

Funkční bloky systému REXYGEN

Referenční příručka

REX Controls s.r.o.

Verze 3.0.4
27.3.2025
Plzeň

Obsah

1	Úvod	17
1.1	Jak číst tuto příručku	17
1.2	Formát popisu funkčních bloků	22
1.3	Konvence pojmenování proměnných, bloků a subsystémů	23
1.4	Kvalita signálu používaná v OPC	25
2	EXEC – Konfigurace exekutivy reálného času	27
	ALARMS – Alarmy systému REXYGEN	29
	ARC – Archiv systému REXYGEN	32
	DATA – Zahrnutí externích souborů	34
	EXEC – Exekutiva reálného času	35
	HMI – Konfigurace vizualizace	37
	INFO – Dodatečné informace o projektu	39
	IODRV – Vstupně-výstupní ovladač systému REXYGEN	40
	IOTASK – Úloha řídicího systému REXYGEN spouštěná ovladačem	42
	LPBRK – Rozpojení zpětné vazby	43
	MODULE – Rozšiřující modul systému REXYGEN	44
	OPCUA – Konfigurace OPCUA serveru systému REXYGEN	45
	OSCALL – Volání funkcí operačního systému	46
	PROJECT – Další nastavení projektu	47
	QTASK – Rychlá úloha řídicího systému REXYGEN	48
	SLEEP – Časovací blok pro Simulink	49
	SRTF – Blok pro nastavování příznaků běhu	50
	STATELOAD – Načtení stavů a parametrů více bloků	52
	STATESAVE – Uložení stavů a parametrů více bloků	54
	SYSEVENT – Čtení systémového logu	56
	SYSLOG – Zápis do systémového logu	58
	TASK – Standardní úloha řídicího systému REXYGEN	59
	TIODRV – Vstupně-výstupní ovladač systému REXYGEN s úlohami	62
	WWW – Obsah pro interní webserver	64

3	INOUT – Bloky vstupů a výstupů	65
	Display – Zobrazení vstupní hodnoty	67
	FromFile – From File	69
	FromWorkspace – From Workspace	70
	GotoTagVisibility – Viditelnost zdroje signálu	71
	INCONN – Blok pro vzdálené získání hodnoty	72
	INQUAD, INOCT, INHEXD – Bloky vícenásobných vstupů	73
	From, INSTD – Připojení signálu nebo vstupní signál	75
	IOASYNC – Asynchronní čtení a zápis	77
	OUTCONN – Blok pro nastavení vzdálené hodnoty	78
	OUTQUAD, OUTOCT, OUTHEXD – Bloky vícenásobných výstupů	79
	OUTRQUAD, OUTROCT, OUTRHEXD – Vícenásobné výstupy s verifikací	82
	OUTRSTD – Výstupní signál s verifikací hodnoty	84
	Goto, OUTSTD – Zdroj signálu nebo výstupní signál	85
	Inport, Outport – Vstupní a výstupní port	87
	QFC – Kódování příznaků kvality signálu	89
	QFD – Dekódování příznaků kvality signálu	90
	SubSystem – Subsystém	91
	ToFile – To File	93
	ToWorkspace – To Workspace	94
	VIN – Ověření kvality vstupního signálu	95
	VOUT – Nastavení kvality výstupního signálu	97
4	MATH – Matematické bloky	99
	ABS – Absolutní hodnota	101
	ADD – Součet dvou signálů	102
	ADDQUAD, ADDOCT, ADDHEXD – Součet více signálů	103
	CNB – Booleovská (logická) konstanta	104
	CNE – Předdefinovaná konstanta	105
	CNI – Celočíselná konstanta	106
	CNR – Reálná konstanta	107
	DIF – Blok difference	108
	DIV – Dělení dvou signálů	109
	EAS – Rozšířené sčítání a odečítání	110
	EMD – Rozšířené násobení a dělení	111
	FNX – Výpočet hodnoty funkce jedné proměnné	112
	FNXY – Výpočet hodnoty funkce dvou proměnných	114
	GAIN – Násobení konstantou	116
	GRADS – Gradientní optimalizace	117
	IADD – Celočíselné sčítání	119
	IDIV – Celočíselné dělení	121
	IMOD – Zbytek po celočíselném dělení	122
	IMUL – Celočíselné násobení	123
	ISUB – Celočíselné odčítání	125

LIN – Lineární interpolace	127
MUL – Násobení dvou signálů	128
NANINF – Blok pro kontrolu hodnot NaN a Inf	129
POL – Vyhodnocení polynomu	131
REC – Převrácená hodnota	132
REL – Relační operace dvou signálů	133
RTOI – Konverze reálného čísla na celé číslo	134
SQR – Druhá mocnina	136
SQRT – Druhá odmocnina	137
SUB – Odčítání dvou signálů	138
UTOI – Konverze celého čísla bez znaménka na celé číslo se znaménkem	139
5 ANALOG – Zpracování analogových signálů	141
ABSROT – Víceotáčkové rozšíření čidla polohy	143
ASW – Přepínač s automatickou volbou vstupu	145
AVG – Filtr: vlečný průměr	147
AVS – Rozběhová jednotka	148
AVSI – Hladká interpolace trajektorie	149
BPF – Filtr: pásmová propust'	152
CMP – Komparátor s hysterezí	153
CNDR – Kompenzátor složité nelinearity	154
DEL – Dopravní zpoždění s inicializací	156
DELM – Dopravní zpoždění	157
DER – Derivace, filtrace a predikce z posledních $n+1$ vzorků	158
EVAR – Vlečná střední hodnota a směrodatná odchylka	159
INTE – Řízený integrátor	160
KDER – Derivace a filtrace vstupního signálu	162
LPF – Filtr: dolní propust'	164
MINMAX – Vlečné minimum a maximum	166
NSCL – Kompenzátor jednoduché nelinearity	167
OSD – Jednokrokové zpoždění	168
RDFT – Vlečná diskrétní Fourierova transformace	169
RLIM – Omezovač strmosti	171
S1OF2 – Výběr jednoho ze dvou analogových vstupů	172
SAI – Zabezpečený analogový vstup	175
SEL – Selektor analogového signálu	178
SELQUAD, SELOCT, SELHEXD – Selektory analogového signálu	179
SHIFTOCT – Posuvný registr pro průběžné ukládání hodnot	181
SHLD – Vzorkovač (sample and hold)	183
SINT – Jednoduchý integrátor	185
SPIKE – Filtr pro potlačení poruch ve tvaru úzkých pulzů	186
SSW – Jednoduchý přepínač	188
SWR – Přepínač s rampovou funkcí	189
VDEL – Dopravní zpoždění s proměnnou délkou	190

ZV4IS – Tvarovač vstupního signálu pro potlačení vibrací	191
6 GEN – Generátory signálů	195
ANLS – Řízený generátor po částech lineární funkce	196
BINS – Řízený generátor binární posloupnosti	198
BIS – Generátor binární posloupnosti	200
BISR – Generátor binární posloupnosti s možností pozastavení	202
MP – Ručně generovaný pulz	204
PRBS – Pseudonáhodná binární posloupnost	205
SG, SGI – Generátory signálu	207
7 REG – Bloky pro regulaci	209
ARLY – Relé s předstihem	211
FIWR – Frekvenční identifikace s rekonstruktorem	212
FLCU – Fuzzy regulátor	217
FRID – Identifikace frekvenční charakteristiky	220
I3PM – Identifikace modelu se třemi parametry	224
LC – Derivační kompenzátor	226
LLC – Integračně-derivační kompenzátor	227
LPI – Hodnocení kvality regulační smyčky	228
MCU – Jednotka pro ruční zadávání	230
PIDAT – PID regulátor s reléovým autotunerem	232
PIDE – PID regulátor se statikou	235
PIDGS – PID regulátor s přepínáním sad parametrů	237
PIDMA – PID regulátor s momentovým autotunerem	239
PIDMAX – PID regulátor s rozšířeným momentovým autotunerem	246
PIDU – PID regulátor	253
PIDUI – PID regulátor s parametry na vstupech	257
POUT – Pulzní výstup	259
PRGM – Programátor	260
PSMPC – Prediktivní regulátor typu pulse-step	262
PWM – Pulzně-šířková modulace	266
RLY – Relé s hystezí	268
SAT – Saturace výstupu s proměnnými mezemi	269
SC2FA – Stavový regulátor systému 2. řádu s autotunerem	271
SCU – Krokový regulátor s polohovou zpětnou vazbou	278
SCUV – Krokový regulátor s rychlostním výstupem	281
SELU – Selektor aktivního regulátoru	285
SMHCC – Regulátor pro procesy s topením a chlazením	287
SMHCCA – Regulátor pro procesy s topením a chlazením s autotunerem	291
SWU – Přepínač vstupu pro vysledování	298
TSE – Třístavový prvek	299

8 LOGIC – Logické řízení	301
AND – Logický součin dvou signálů	302
ANDQUAD, ANDOCT, ANDHEXD – Logický součin více signálů	303
ATMT – Automat pro sekvenční řízení	304
BDOCT, BDHEXD – Bitové demultiplexery	306
BITOP – Bitová operace dvou celočíselných signálů	307
BMOCT, BMHEXD – Bitové multiplexery	308
COUNT – Řízený čítač	309
EATMT – Rozšířený automat pro sekvenční řízení	311
EDGE – Detekce hrany logického signálu	314
EQ – Shodnost dvou signálů	315
INTSM – Bitový posun a maska nad celým číslem	316
ISSW – Jednoduchý přepínač celočíselných signálů	317
ITOI – Transformace celých a binárních čísel	318
NOT – Logická negace	319
OR – Logický součet dvou signálů	320
ORQUAD, OROCT, ORHEXD – Logický součet více signálů	321
RS – Klopný obvod	322
SR – Klopný obvod	323
TIMER – Vícefunkční časovač	324
9 TIME – Bloky pro práci s časem	327
DATE – Aktuální datum	328
DATETIME – Čtení, nastavování a konverze času	329
TC – Řízení časovače	332
TIME – Aktuální čas	334
TS – Aktuální časová značka	335
TS2NS – Rozdíl časových značek v nanosekundách	337
WSCH – Týdenní programátor	338
10 ARC – Archivace dat	341
ACD – Archivní komprese s použitím delta kritéria	344
ACLEAR – Vynucené smazání archivu	346
AFLUSH – Vynucené zapsání archivu	347
ALB, ALBI – Alarmy pro logickou hodnotu	348
ALM, ALMI – Uložení hodnoty do alarmu	350
ALN, ALNI – Alarmy pro číselnou hodnotu	351
ARS – Uložení hodnoty do archivu	354
TRND – Záznam trendů v reálném čase	356
TRNDV – Záznam trendů v reálném čase (pro vektory)	359

11 STRING – Bloky pro práci s řetězci	361
CNS – Textová konstanta	362
CONCAT – Spojení stringů (podle vzoru)	363
FIND – Nalezení textu	364
ITOS – Konverze celého čísla na text	365
LEN – Délka textu	366
MID – Výřez textu	367
PJROCT – Získání číselných hodnot z textu ve formátu JSON	368
PJSEXOCT – Získání textových hodnot z textu ve formátu JSON	370
PJSOCT – Získání textových hodnot z textu ve formátu JSON	371
REGEXP – Parser regulárních výrazů	373
REPLACE – Náhrada textu	376
RTOS – Konverze čísla na text	377
SELSOCT – Výběr textu z několika vstupů	378
STOR – Konverze textu na číslo	380
TRIM – Odstranění mezer na začátku a konci textu	381
12 PARAM – Bloky pro manipulaci s parametry	383
GETPA – Blok pro vzdálené získání vektorového parametru	384
GETPB, GETPI, GETPR – Bloky pro vzdálené získání parametru	386
GETPS – Blok pro vzdálené získání parametru typu string	388
GETPX – Blok pro vzdálené získání parametru	389
PARA – Blok s vektorovým parametrem nastavitelným ze vstupu	391
PARE – Blok s parametrem výběr ze seznamu nastavitelným ze vstupu	393
PARB, PARI, PARR – Bloky s nastavitelným parametrem ze vstupu	394
PARS – Blok s parametrem typu string nastavitelným ze vstupu	396
PARX – Blok s parametrem nastavitelným ze vstupu	397
SETPA – Blok pro vzdálené nastavování vektorového parametru	399
SETPB, SETPI, SETPR – Bloky pro vzdálené nastavování parametru	401
SETPS – Blok pro vzdálené nastavování parametru typu string	404
SETPX – Blok pro vzdálené nastavování parametru	405
SGSLP – Nastavování, čtení, ukládání a načítání parametrů	407
SILO – Uložení vstupního signálu, načtení výstupního signálu	411
SILOS – Uložení vstupního řetězce, načtení výstupního řetězce	413
13 MODEL – Simulace dynamických systémů	415
CDELSSM – Stavový model spojitého lineárního systému s dopravním zpožděním	417
CSSM – Stavový model spojitého lineárního systému	420
DDELSSM – Stavový model diskrétního lineárního systému s dopravním zpožděním	423
DFIR – Filtr s konečnou impulzní odezvou	425
DSSM – Stavový model diskrétního lineárního systému	426
EKF – Rozšířený (nelineární) Kalmanův filtr	428

FOPDT – Model systému 1. řádu s dopravním zpožděním	431
IPEN2, IPEN3 – N-ramenné inverzní kyvadlo na vozíku - Fyzikální parametry	432
IPEN2pu, IPEN3pu – N-ramenné inverzní kyvadlo na vozíku - Dynamické parametry	435
MDL – Model procesu	437
MDLI – Model procesu s proměnnými parametry	438
MVD – Motorizovaný pohon ventilu	439
NSSM – Nelineární stavový model	440
NUREACT – Model of nuclear reactor	443
QCOPT – Model dronu (kvadrukoptéry)	444
SGEN – Model synchronního generátoru	446
SGENTX – Model synchronního generátoru	448
SOPDT – Model systému 2. řádu s dopravním zpožděním	450
STMGEN – Model parogenerátoru	452
STURB – Model parní turbíny	454
14 MATRIX – Bloky pro maticové a vektorové operace	457
CNA – Konstantní pole (vektor/matice)	460
MB_DASUM – Součet absolutních hodnot	461
MB_DAXPY – Provádí $y := a*x + y$ pro vektory x, y	463
MB_DCOPY – Kopíruje vektor x do vektoru y	465
MB_DDOT – Skalární součin dvou vektorů	467
MB_DGEMM – Provádí $C := \alpha*op(A)*op(B) + \beta*C$, where $op(X) = X$ or $op(X) = X^T$	469
MB_DGEMV – Provádí $y := \alpha*A*x + \beta*y$ or $y := \alpha*A^T*x + \beta*y$	471
MB_DGER – Provádí $A := \alpha*x*y^T + A$	473
MB_DNRM2 – Eukleidovská norma vektoru	475
MB_DROT – Rovinná rotace vektoru	477
MB_DSCAL – Násobení vektoru konstantou	479
MB_DSWAP – Záměna dvou vektorů	481
MB_DTRMM – Provádí $B := \alpha*op(A)*B$ or $B := \alpha*B*op(A)$, where $op(X) = X$ or $op(X) = X^T$ pro trojúhelníkovou matici A	483
MB_DTRMV – Provádí $x := A*x$ or $x := A^T*x$ pro trojúhelníkovou matici A	485
MB_DTRSV – Řeší jednu ze soustav rovnic $A*x = B$ nebo $A^T*x = B$ pro trojúhelníkovou matici A	487
ML_DGEBAK – Zpětná transformace k ML_DGEBAL levých nebo pravých vlastních vektorů	489
ML_DGEBAL – Vyvážení obecné reálné matice	492
ML_DGEBRD – Redukce obecné reálné matice do bidiagonální formy pomocí ortogonální transformace	494
ML_DGECON – Odhad převrácené hodnoty čísla podmíněnosti obecné reálné matice	496

ML_DGEES – Výpočet vlastních čísel, Schurovy formy a volitelně matice Schurových vektorů	498
ML_DGEEV – Výpočet vlastních čísel a volitelně levých a/nebo pravých vlastních vektorů	500
ML_DGEHRD – Redukce reálné obecné matice A na horní Hessenbergovu formu	502
ML_DGELQF – Výpočet LQ faktorizace reálné matice A s rozměry M x N .	505
ML_DGELSD – Výpočet řešení s minimální normou reálné lineární úlohy nejmenších čtverců	507
ML_DGEQRF – Výpočet QR faktorizace reálné matice A s rozměry M x N .	509
ML_DGESDD – Výpočet singulární dekompozice (SVD) reálné matice A s rozměry M x N	511
ML_DLACPY – Kopíruje celou nebo část matice do jiné matice	513
ML_DLANGE – Výpočet některé z maticových norem obecné matice	515
ML_DLASET – Inicializuje mimodiagonální a diagonální prvky matice na zadané hodnoty	517
ML_DTRSYL – Řešení reálné Sylvesterovy rovnice pro kvazitrojúhelníkové matice A a B	519
MX_AT – Hodnota prvku matice/vektoru	521
MX_ATSET – Nastavení hodnoty prvku matice/vektoru	522
MX_CNADD – Přičte skalár ke každému prvku matice/vektoru	523
MX_CNMUL – Vynásobí matici/vektor skalárem	524
MX_CTODPA – Diskretizace spojitého modelu (A,B) do (Ad,Bd) s využitím Padéových aproximací	525
MX_DIM – Dimenze matice/vektoru	527
MX_DIMSET – Nastavení dimenze matice/vektoru	528
MX_DSAGET – Uložení submatice A do matice B	530
MX_DSAREF – Nastavení odkazu na submatici A do matice B	532
MX_DSASET – Uložení matice A do submatice v B	534
MX_DTRNSP – Transpozice obecné matice: $B := \alpha * A^T$	536
MX_DTRNSQ – Transpozice čtvercové matice na místě: $A := \alpha * A^T$. .	538
MX_FILL – Vyplnění reálné matice/vektoru	540
MX_FNX – Maticové a vektorové skalární funkce	541
MX_MAT – Blok pro uložení dat matice	543
MX_RAND – Náhodně vygenerovaná matice nebo vektor	544
MX_REFCOPY – Kopírování vstupních odkazů na matice A a B do jejich výstupních odkazů	546
MX_SLFS – Ukládání a čtení matice/vektoru do souboru nebo textového řetězce	547
MX_VEC – Blok pro uložení dat vektoru	550
MX_WRITE – Výpis matice/vektoru do konzole/systémového logu	551
RTOV – Vektorový multiplexer	553
SWVMR – Přepínač vektorového/maticového/odkazovacího signálu	555
VTOR – Vektorový demultiplexer	556

15 OPTIM – Bloky pro optimalizaci	557
QP_MPC2QP – Převod úlohy prediktivního řízení na kvadratické programování	558
QP_OASES – Kvadratické programování pomocí metody aktivní množiny . . .	565
QP_UPDATE – Aktualizace matic/vektorů kvadratického programování . . .	570
SOLNP – Solver pro nelineární optimalizaci	575
16 SPEC – Speciální bloky	579
EPC – Spouštění externích programů	580
HTTP – Blok pro generování požadavků HTTP GET a POST (zastaralý) .	583
HTTP2 – Blok pro generování požadavků HTTP	585
RDC – Komunikační blok	587
REXLANG – Volně programovatelný blok	590
SMTP – Blok pro odesílání e-mailových oznámení přes SMTP	611
STEAM – Přepoččet vlastností páry	613
UART – Komunikační blok pro UART	616
17 LANG – Jazykové bloky	619
LUA, LUAQUAD, LUAOCT, LUAHEXD – Volně programovatelné bloky v Lua . . .	620
PYTHON – Volně programovatelný blok v jazyce Python	624
18 DSP – Zpracování speciálních signálů	631
BSFIFO – Binární Struktura - serializace a deserializace do cyklického bufferu	632
BSGET, BSGETOCT – Binární struktura - získání hodnoty daného typ	634
BSGETV, BSGETOCTV – Binární struktura - získání pole hodnot daného typu	636
BSSET, BSSETOCT – Binární struktura - nastavení hodnoty daného typu . .	637
BSSETV, BSSETOCTV – Binární struktura - nastavení pole hodnot daného typu	638
FFT – Rychlá Fourierova transformace	639
MOSS – Přesný senzor pohybu	641
PCI – Přístup k paměti přes sběrnici PCI	643
PSD – Spektrální výkonová hustota	644
RAWCOPY – Kopírování vektoru po 1, 2 nebo 4 Bajtech	650
19 MQTTDrv – Komunikace přes MQTT protokol	651
MqttPublish – Odeslání zprávy protokolem MQTT	652
MqttSubscribe – Odběr zpráv z MQTT topic	654
20 MC_SINGLE – Řízení pohybu v jedné ose	657
MCP_AccelerationProfile – * Generování trajektorie (zrychlení)	660
MCP_Halt – * Zastavení pohybu (přerušitelné)	662
MCP_HaltSuperimposed – * Zastavení pohybu (přídavné a přerušitelné) . .	663
MCP_Home – * Nalezení výchozí polohy	664
MCP_MoveAbsolute – * Pohyb do pozice (absolutní souřadnice)	666

MCP_MoveAdditive – * Pohyb do pozice (relativně ke konci předchozího pohybu)	668
MCP_MoveContinuousAbsolute – * Pohyb do pozice (absolutní souřadnice)	670
MCP_MoveContinuousRelative – * Pohyb do pozice (relativně ke konci předchozího pohybu)	672
MCP_MoveRelative – * Kartézské transformace	674
MCP_MoveSuperimposed – * Pohyb do pozice (přídavný pohyb)	676
MCP_MoveVelocity – * Pohyb konstantní rychlostí	677
MCP_PositionProfile – * Generování trajektorie (poloha)	679
MCP_SetOverride – * Nastavení násobivých faktorů na ose	681
MCP_Stop – * Zastavení pohybu	682
MCP_TorqueControl – * Řízení síly/momentu	683
MCP_VelocityProfile – * Generování trajektorie (rychlost)	685
MC_AccelerationProfile, MCP_AccelerationProfile – Generování trajektorie (zrychlení)	687
MC_Halt, MCP_Halt – Zastavení pohybu (přerušitelné)	691
MC_HaltSuperimposed, MCP_HaltSuperimposed – Zastavení pohybu (přídavné a přerušitelné)	693
MC_Home, MCP_Home – Nalezení výchozí polohy	694
MC_MoveAbsolute, MCP_MoveAbsolute – Pohyb do pozice (absolutní souřadnice)	696
MC_MoveAdditive, MCP_MoveAdditive – Pohyb do pozice (relativně ke konci předchozího pohybu)	699
MC_MoveContinuousAbsolute, MCP_MoveContinuousAbsolute – Pohyb do pozice (absolutní souřadnice)	702
MC_MoveContinuousRelative, MCP_MoveContinuousRelative – Pohyb do pozice (relativně ke konci předchozího pohybu)	706
MC_MoveRelative, MCP_MoveRelative – Pohyb do pozice (relativně k okamžiku spuštění)	710
MC_MoveSuperimposed, MCP_MoveSuperimposed – Pohyb do pozice (přídavný pohyb)	713
MC_MoveVelocity, MCP_MoveVelocity – Pohyb konstantní rychlostí	716
MC_PositionProfile, MCP_PositionProfile – Generování trajektorie (poloha)	719
MC_Power – Aktivace osy	723
MC_ReadActualPosition – Skutečná poloha osy	724
MC_ReadAxisError – Chyba osy	725
MC_ReadBoolParameter – Čtení parametru (logická hodnota)	726
MC_ReadParameter – Čtení parametru (číselná hodnota)	727
MC_ReadStatus – Stav osy	729
MC_Reset – Nulování chyb osy	731
MC_SetOverride, MCP_SetOverride – Nastavení násobivých faktorů na ose	732
MC_Stop, MCP_Stop – Zastavení pohybu	734
MC_TorqueControl, MCP_TorqueControl – Řízení síly/momentu	736

MC_VelocityProfile, MCP_VelocityProfile – Generování trajektorie (rychlost)	739
MC_WriteBoolParameter – Nastavení parametru (logická hodnota)	743
MC_WriteParameter – Nastavení parametru (číselná hodnota)	744
RM_Axis – Osa pro řízení pohybu	745
RM_AxisOut – Výstupní blok osy	752
RM_AxisSpline – Interpolace požadované polohy (rychlosti, zrychlení)	753
RM_HomeOffset – * Nastavení počáteční polohy pomocí offsetu	758
RM_Track – Sledování a krokování	759
21 MC _MULTI – Řízení pohybu více os	761
MCP_CamIn – * Zapnutí vačky	763
MCP_CamTableSelect – Definice vačky	765
MCP_CombineAxes – * Kombinace pohybu dvou os do třetí	767
MCP_GearIn – * Zapnutí konstantního převodového poměru	769
MCP_GearInPos – * Zapnutí konstantního převodového poměru v zadané pozici	771
MCP_PhasingAbsolute – * Vytvoření fázového posunu (absolutní souřadnice)	773
MCP_PhasingRelative – * Vytvoření fázového posunu (relativně k pozici při spuštění)	775
MC_CamIn, MCP_CamIn – Zapnutí vačky	777
MC_CamOut – Vypnutí vačky	781
MC_CombineAxes, MCP_CombineAxes – Kombinace pohybu dvou os do třetí	783
MC_GearIn, MCP_GearIn – Zapnutí konstantního převodového poměru	786
MC_GearInPos, MCP_GearInPos – Zapnutí konstantního převodového poměru v zadané pozici	789
MC_GearOut – Vypnutí konstantního převodového poměru	794
MC_PhasingAbsolute, MCP_PhasingAbsolute – Vytvoření fázového posunu (absolutní souřadnice)	796
MC_PhasingRelative, MCP_PhasingRelative – Vytvoření fázového posunu (relativně k pozici při spuštění)	799
22 MC _COORD – Koordinované řízení pohybu	801
MCP_GroupHalt – * Zastavení koordinovaného pohybu (přerušitelné)	805
MCP_GroupInterrupt – * Přerušení pohybu skupiny os	806
MCP_GroupSetOverride – * Nastavení násobivých faktorů na osách ve skupině	807
MCP_GroupSetPosition – * Nastavení polohového offsetu skupiny os	808
MCP_GroupStop – * Zastavení koordinovaného pohybu	809
MCP_MoveCircularAbsolute – * Pohyb do pozice po kružnicích (absolutní souřadnice)	810
MCP_MoveCircularRelative – * Pohyb do pozice po kružnicích (relativní souřadnice)	812

MCP_MoveDirectAbsolute – * Nekoordinovaný pohyb do pozice (absolutní souřadnice)	814
MCP_MoveDirectRelative – * Nekoordinovaný pohyb do pozice (relativní souřadnice)	816
MCP_MoveLinearAbsolute – * Pohyb do pozice po přímkách (absolutní souřadnice)	818
MCP_MoveLinearRelative – * Pohyb do pozice po přímkách (relativní souřadnice)	820
MCP_MovePath – * Generování obecné trajektorie v prostoru	822
MCP_MovePath_PH – * Generování obecné trajektorie v prostoru PH	824
MCP_SetCartesianTransform – * Kartézské transformace	826
MCP_SetKinTransform_Arm – * Kinematické transformace robot ARM	827
MCP_SetKinTransform_UR – * Kinematická transformace pro UR robot	829
MC_AddAxisToGroup – Přidání osy do skupiny os	831
MC_GroupContinue – Pokračování v přerušném pohybu	832
MC_GroupDisable – Převedení skupiny do stavu GroupDisabled	833
MC_GroupEnable – Převedení skupiny do stavu GroupStandby	834
MC_GroupHalt, MCP_GroupHalt – Zastavení koordinovaného pohybu (přerušitelné)	835
MC_GroupInterrupt, MCP_GroupInterrupt – Přerušování pohybu skupiny os	841
MC_GroupReadActualAcceleration – Aktuální zrychlení skupiny os	842
MC_GroupReadActualPosition – Aktuální poloha skupiny os	843
MC_GroupReadActualVelocity – Aktuální rychlost skupiny os	844
MC_GroupReadError – Chyby ve skupině os	845
MC_GroupReadStatus – Stav skupin os	846
MC_GroupReset – Nulování chyb os ve skupině	848
MC_GroupSetOverride, MCP_GroupSetOverride – Nastavení násobivých faktorů na osách ve skupině	849
MC_GroupSetPosition, MCP_GroupSetPosition – Nastavení polohového offsetu skupiny os	852
MC_GroupStop, MCP_GroupStop – Zastavení koordinovaného pohybu	854
MC_MoveCircularAbsolute, MCP_MoveCircularAbsolute – Pohyb do pozice po kružnicích (absolutní souřadnice)	857
MC_MoveCircularRelative, MCP_MoveCircularRelative – Pohyb do pozice po kružnicích (relativní souřadnice)	862
MC_MoveDirectAbsolute, MCP_MoveDirectAbsolute – Nekoordinovaný pohyb do pozice (absolutní souřadnice)	867
MC_MoveDirectRelative, MCP_MoveDirectRelative – Nekoordinovaný pohyb do pozice (relativní souřadnice)	871
MC_MoveLinearAbsolute, MCP_MoveLinearAbsolute – Pohyb do pozice po přímkách (absolutní souřadnice)	875
MC_MoveLinearRelative, MCP_MoveLinearRelative – Pohyb do pozice po přímkách (relativní souřadnice)	879
MC_MovePath, MCP_MovePath – Generování obecné trajektorie v prostoru	883

MC_MovePath_PH – * Generování obecné trajektorie v prostoru PH	886
MC_ReadCartesianTransform – Přechzení použité kartézské transformace	888
MC_SetCartesianTransform, MCP_SetCartesianTransform – Kartézská transformace	890
MC_UngroupAllAxes – Odebrání všech os ze skupiny	893
RM_AxesGroup – Skupina os pro koordinované řízení pohybu	894
RM_Feed – MC „krmič“	899
RM_Gcode – CNC řízení pohybu	901
RM_GroupTrack – * Sledování polohy/rychlosti	904
23 CanDrv – Komunikace po sběrnici CAN	907
CanItem – Další přijatá zpráva sběrnice CAN	908
CanRecv – Přijetí zprávy sběrnice CAN	909
CanSend – Odeslání zprávy na sběrnici CAN	911
24 OpcUaDrv – Komunikace pomocí OPC UA	913
OpcUaReadValue – Čtení hodnoty protokolem OPC UA	914
OpcUaServerValue – Vystavení hodnoty v podobě OPC UA uzlu	916
OpcUaWriteValue – Zápis hodnoty protokolem OPC UA	918
25 UNIPi – Komunikační bloky pro Unipi	921
IM201CNT – Iris IM201DI, 4 digitální čítače	923
IM201DI – Iris IM201DI, 4 digitální vstupy	924
IM203DO – Iris IM203DO, 8 digitálních výstupů	925
IM203PWM – Iris IM203PWM, 8 PWM výstupů	926
IM204CNT – Iris IM204CNT, 16 digitálních čítačů	927
IM204DI – Iris IM204DI, 16 digitálních vstupů	928
IM205DO – Iris IM205DO, 16 digitálních výstupů	929
IM205PWM – Iris IM205PWM, 16 digitálních výstupů	930
IM301CNT – Iris IM301CNT, 2 digitální čítače	932
IM301DI – Iris IM301DI, 2 digitální vstupy	933
IM301DO – Iris IM301DO, 2 digitální výstupy	934
IM502AO – Iris IM502AO, 4 analogové výstupy	935
IM503AI – Iris IM503AI, 8 analogových vstupů	936
IM504RI – Iris IM504RI, 8 odporových nebo teplotních vstupů	938
IM506AI – Iris IM506AI, 2 analogové vstupy	940
IM506AO – Iris IM506AO, 1 analogový výstup	942
IRIS_MODULE – Iris - Informace z popisu modulu	943
UNIPi_CHANNEL – Informace o Iris modulu nebo Patron sekci	944
UNIPi_PRODUCT – Informace z popisu produktu	946
UNIPi_S1AI – Patron sekce 1, analogový vstup	947
UNIPi_S1AOR – Patron sekce 1, analogový výstup nebo odporový vstup	948
UNIPi_S1CNT – Patron sekce 1, čítače	949
UNIPi_S1DI – Patron sekce 1, digitální vstupy	950

UNIPi_S1D0 – Patron sekce 1, digitální výstupy	951
UNIPi_S1LED – Patron sekce 1, LED výstupy	952
UNIPi_S1PWM – Patron sekce 1, PWM výstupy	953
UNIPi_S2AI – Patron sekce 2, analogové vstupy	954
UNIPi_S2AO – Patron sekce 2, analogové výstupy	955
UNIPi_S2CNT – Patron sekce 2, čítače	956
UNIPi_S2DI – Patron sekce 2, digitální vstupy	957
UNIPi_S2RO – Patron sekce 2, reléové výstupy	958
A Licenční politika	959
B Seznam funkčních bloků a jejich licencování	963
C Chybové kódy systému REXYGEN	977
D Speciální signály systému REXYGEN	983
Literatura	985
Rejstřík	987

Poznámka: U bloků označených * je k dispozici pouze částečná dokumentace. Kompletní dokumentace může být dostupná v ostatních jazykových mutacích manuálu.

Kapitola 1

Úvod

Příručka „Funkční bloky systému REXYGEN“ je, jak už její název napovídá, referenční příručkou knihovny RexLib funkčních bloků řídicího systému REXYGEN. Kromě referenčního popisu jednotlivých tříd, popisuje (referenčním způsobem) všechny subsystémy řídicího systému REXYGEN.

1.1 Jak číst tuto příručku

Standardně dodávaná rozsáhlá knihovna funkčních bloků RexLib řídicího systému REXYGEN je rozdělena do menších skupin logicky příbuzných bloků, tzv. *kategorií* (podknihoven). Každá kategorie je popisována v samostatné kapitole, obsahující nejprve obecné vlastnosti celé kategorie a jejích funkčních bloků, následované postupně popisem všech funkčních bloků dané kategorie.

Jednotlivé kapitoly příručky obsahují:

1 Úvod

Tato úvodní kapitola, seznamující s uspořádáním příručky a uvádějící formát (konvenci) popisu jednotlivých funkčních bloků.

2 EXEC – Konfigurace exekutivy reálného času

Knihovna EXEC je nezbytná pro nastavení exekutivy v reálném čase v systému REXYGEN a zahrnuje klíčové bloky jako **EXEC**, **TASK**, **QTASK** a **HMI**. Tyto bloky jsou fundamentální pro správu provádění úkolů, určování priority procesů a pro interakci s rozhraními pro uživatele, čímž podstatně přispívají k efektivitě a ovladatelnosti aplikací v ekosystému REXYGEN.

3 INOUT – Bloky vstupů a výstupů

Knihovna INOUT slouží jako klíčové rozhraní v systému REXYGEN, umožňuje plynulou interakci s ovladači vstupů/výstupů. Je navržena pro efektivní simultánní zpracování signálů, což je zásadní pro úlohy rychlého řízení. Tato knihovna zjednodušuje propojení mezi řídicími algoritmy a hardwarem, zajišťuje minimální

latenci. Zároveň poskytuje pokročilé funkce, jako jsou virtuální propojení (vlajky) signálů pro zvýšenou přehlednost schémat a flexibilitu podsystémů.

4 MATH – Matematické bloky

Knihovna MATH nabízí ucelenou sbírku matematických operací a funkcí. Obsahuje základní aritmetické bloky jako **ADD**, **SUB**, **MUL** a **DIV** pro standardní výpočty a specializovanější bloky, jako je **ABS** pro absolutní hodnoty, **SQRT** pro odmocniny a **SQR** pro odmocnění. Pokročilé funkce poskytují bloky jako **LIN** pro lineární transformace, **POL** pro vyhodnocení polynomů a **FNX**, **FNXY** pro přizpůsobitelné matematické funkce. Knihovna také nabízí operace specifické pro celá čísla prostřednictvím bloků jako **IADD**, **IMUL**, **IDIV** a **IMOD**.

5 ANALOG – Zpracování analogových signálů

Knihovna představuje univerzální škálu funkčních bloků určených pro aplikace řízení a zpracování signálů. Obsahuje bloky jako **ASW**, **AVG**, **BPF** a **DEL**, které nabízejí funkce od manipulace se signálem a průměrování po filtrování a složité podmíněné operace, čímž pokrývají široké spektrum požadavků a scénářů.

6 GEN – Generátory signálů

Knihovna GEN je specializována na generování signálů. Obsahuje bloky jako **ANLS** pro generování po částech lineární funkce času nebo generátory binárních sekvencí **BINS**, **BIS**, **BISR**. Knihovna také nabízí **MP** pro manuální generování pulsních signálů, **PRBS** pro generování pseudonáhodných binárních sekvencí a **SG** pro generování periodických signálů. Tato knihovna poskytuje zásadní nástroje pro vytváření a manipulaci s různými typy signálů.

7 REG – Bloky pro regulaci

Bloky pro regulaci tvoří nejrozsáhlejší podknihovnu knihovny RexLib a zahrnují bloky od jednoduchých dynamických kompenzátorů, přes bloky pro přepínání regulačních struktur, bloky pro přizpůsobení výstupů akčním členům (krokové regulátory, šířková modulace) až po několik verzí PID (P, I, PI, PD a PID) regulátorů. Mezi regulátory jsou např. blok **PIDGS**, umožňující za běhu přepínat několik sad parametrů (tzv. *gain scheduling*), **PIDMA** s vestavěným *momentovým autotunerem*, blok **PIDAT** s vestavěným reléovým autotunerem nebo blok fuzzy regulátoru **FLCU**, a další.

8 LOGIC – Logické řízení

Knihovna LOGIC zahrnuje řadu bloků pro provádění logických a sekvenčních operací. Zahrnuje základní booleovské bloky jako **AND**, **OR**, **NOT** pro základní logické operace a pokročilé bloky jako **ATMT** pro konečné automaty. Bloky jako **COUNT** a **TIMER** rozšiřují funkčnost na obousměrné počítání pulsů a operace založené na čase. Další prvky jako **BITOP**, **BMOCT** a **BDOCT** nabízejí bitové operace a možnosti multiplexování/demultiplexování, čímž zvyšují všestrannost knihovny při ovládání kombinační a sekvenční logiky.

9 TIME – Bloky pro práci s časem

Knihovna TIME se specializuje na časově založené operace a plánování v systému REXYGEN. Obsahuje bloky jako **DATE**, **TIME** a **DATETIME** pro manipulaci s datem a poskytuje základní nástroje pro práci s časovými daty. Knihovna obsahuje **TC** pro ovládání vnitřního časovače. Kromě toho se pro plánování používá **WSCH**, což umožňuje efektivní správu časově závislých úloh. Tato knihovna je zvláště cenná pro systémy vyžadující přesné řízení času a možnosti plánování.

10 ARC – Archivace dat

Exekutiva systému REXYGEN se skládá z různých propojených podsystémů (podsystém v reálném čase, diagnostický podsystém, podsystém ovladačů atd.). Jeden z těchto podsystémů je *archivační podsystém*. Archivační podsystém se stará o zaznamenávání historie řídicího algoritmu.

11 STRING – Bloky pro práci s řetězci

Knihovna STRING je věnována manipulaci a analýze řetězců v systému REXYGEN. Zahrnuje bloky jako **CONCAT** pro zřetězení řetězců, **FIND** pro vyhledávání v řetězcích a **REPLACE** pro nahrazování segmentů řetězců. Knihovna nabízí **LEN** a **MID** pro určení délky řetězce a extrakci podřetězců. Pokročilé porovnávání vzorů poskytuje **REGEXP**. Konverzní bloky jako **ITOS**, **STOR** a **RTOS** převádějí celá a reálná čísla na řetězce, zatímco jednoduchý blok **CNS** definuje řetězcovou konstantu. Knihovna navíc obsahuje bloky jako **PJROCT** pro analýzu JSON. Tato kolekce bloků je nezbytná pro manipulaci a zpracování řetězcových dat v různých aplikacích.

12 PARAM – Bloky pro manipulaci s parametry

Knihovna PARAM je určena pro správu parametrů a zpracování signálů v systému REXYGEN. Obsahuje bloky jako **PARR** a jeho varianty pro definování a úpravu různých typů parametrů. Bloky pro čtení parametrů jiných bloků jako **GETPA** a **GETPS**. Naopak **SETPA**, **SETPR** a **SETPS** se používají pro dynamické nastavení hodnot parametrů jiných bloků. Knihovna navíc obsahuje **SILO** a **SILOS** pro export a import hodnot ze souboru. Tato knihovna je klíčová pro systémy vyžadující dynamickou manipulaci s parametry a možnosti čtení/ukládání hodnot do souboru.

13 MODEL – Simulace dynamických systémů

Knihovna MODEL je zaměřena na modelování a simulaci systému. Zahrnuje bloky jako **CSSM** a **DSSM** pro spojitě a diskrétní modely stavového prostoru a **DFIR** pro digitální filtry s konečnou impulsní odezvou. Knihovna nabízí **EKF** pro implementaci rozšířeného Kalmanova filtru a **FOPDT**, **SOPDT** pro modely časového zpoždění procesu prvního a druhého řádu. Navíc poskytuje **FMUCS** a **FMUINFO** pro propojení s funkčními mock-upovými jednotkami a **MDL**, **MDLI** pro rozhraní generických modelů. Pokročilé funkce jsou pokryty bloky jako **CDELSSM**, **DDELSSM** pro stavové modely kontinuálních a diskrétních systémů se zpožděním a **MVD** pro modelové proměnné zpoždění, které splňují širokou škálu požadavků na modelování v systému REXYGEN.

14 MATRIX – Bloky pro maticové a vektorové operace

Knihovna MATRIX je navržena pro pokročilé maticové výpočty a manipulace. Zahrnuje širokou škálu bloků, jako je [MB_DGEMM](#), [MB_DTRMM](#) a [MB_DGER](#) pro operace typu matice-matice a matice-vektor. Knihovna obsahuje funkce pro rozklad matic ([ML_DGEBRD](#), [ML_DGEQRF](#)), problémy s vlastními hodnotami ([ML_DGEEV](#), [ML_DGEES](#)) a rozklad singulární hodnoty ([ML_DGESDD](#)). Kromě toho nabízí pomocné bloky jako [MX_MAT](#), [MX_VEC](#) a [MX_FILL](#) pro vytváření matic a manipulaci, stejně jako specializované bloky, jako je např. [MX_DTRNSP](#) pro maticovou transpozici a [MX_RAND](#) pro generování náhodných matic. Tato knihovna je nezbytná pro složité matematické operace zahrnující matice v různých aplikacích.

15 OPTIM – Bloky pro optimalizaci

Knihovna OPTIM je přizpůsobena pro optimalizační algoritmy a procesy. Zahrnuje [QCEDPOPT](#) pro optimalizace tzv. Economic Dispatch problémů, poskytující pokročilé nástroje pro řešení složitých optimalizačních problémů. Knihovna také obsahuje bloky jako [QP_MPC2QP](#) a [QP_OASES](#) pro kvadratické programování, které jsou nezbytné ve scénářích Model Predictive Control (MPC). Navíc je k dispozici [QP_UPDATE](#) pro aktualizaci parametrů kvadratického programu. Tato knihovna je zvláště užitečná v systémech vyžadujících řešení optimalizace na vysoké úrovni, jako jsou pokročilé řídicí a rozhodovací algoritmy.

16 SPEC – Speciální bloky

Knihovna SPEC zahrnuje rozmanitou sadu funkčních bloků navržených pro integraci široké škály funkcí do automatizace, řídicích systémů a komunikačních protokolů. Od usnadnění přesných termodynamických výpočtů s blokem [STEAM](#) po umožnění bezproblémové datové komunikace prostřednictvím [UART](#) a [SMTP](#) slouží knihovna jako komplexní sada nástrojů pro inženýry a vývojáře. Obsahuje specializované bloky pro spouštění externích programů ([EPC](#)), zpracování webových požadavků ([HTTP2](#)). Navíc nabízí unikátní vstupně-výstupní řešení ([RDC](#)) a všestranné programovací prostředí s [REXLANG](#).

17 LANG – Jazykové bloky

Standardní funkční bloky systému REXYGEN pokrývají nejtypičtější potřeby v řídicích aplikacích. Nicméně stále existují situace, kdy je nutné (nebo pohodlnější) implementovat uživatelsky definovanou funkci. Pro tyto účely slouží bloky z knihovny LANG, případně blok [REXLANG](#).

18 DSP – Zpracování speciálních signálů

Knihovna DSP je přizpůsobena pro pokročilé zpracování digitálních signálů. Obsahuje bloky jako [FFT](#) pro operace rychlé Fourierovy transformace a [PSD](#) pro analýzu spektrální hustoty výkonu. Knihovna také nabízí [BSFIFO](#), [BSGET](#), [BSGETV](#), [BSSET](#) a [BSSETV](#) pro ukládání do bufferu a jeho načítání, což umožňuje efektivní manipulaci s daty při úlohách zpracování signálů. Kromě toho knihovna obsahuje blok [MOSS](#) - pokročilý filtr pro inkrementální senzory. Tato kolekce bloků je nezbytná pro sofistikovanou analýzu a manipulaci se signály v digitálních systémech.

19 MQTTDrv – Komunikace přes MQTT protokol

Knihovna MQTTDrv je určena pro komunikaci IoT (Internet of Things) pomocí protokolu MQTT (Message Queuing Telemetry Transport). Skládá se ze dvou primárních bloků: `MqttPublish` a `MqttSubscribe`. Blok `MqttPublish` se používá pro odesílání zpráv brokerovi MQTT, umožňující publikování dat do MQTT témat. Naopak blok `MqttSubscribe` je určen pro přihlášení k odběru témat a příjem zpráv od brokera. Tato knihovna usnadňuje efektivní a efektivní datovou komunikaci v aplikacích IoT a využívá lehký a široce používaný protokol MQTT pro výměnu zpráv.

20 MC_SINGLE – Řízení pohybu v jedné ose

Knihovna MC_SINGLE je určena pro řízení pohybu v jednoosých systémech. Obsahuje bloky jako `MC_MoveAbsolute`, `MC_MoveRelative` a `MC_MoveVelocity` pro přesné polohování a ovládání rychlosti. Knihovna obsahuje `MC_Home` pro operace navádění a `MC_Power` pro řízení stavu napájení osy. Pokročilé funkce poskytují `MC_AccelerationProfile`, `MC_PositionProfile` a `MC_VelocityProfile` pro přizpůsobení profilů pohybu. Nabízí také možnosti monitorování a úpravy parametrů prostřednictvím `MC_ReadActualPosition`, `MC_ReadAxisError`, `MC_ReadParameter` a `MC_WriteParameter`. Knihovna navíc obsahuje bloky jako `MC_Halt`, `MC_Reset` a `MC_Stop` pro nouzové a kontrolní operace. Tato knihovna je nezbytná pro aplikace vyžadující přesný a řízený pohyb v jednoosých konfiguracích.

21 MC_MULTI – Řízení pohybu více os

Knihovna MC_MULTI se specializuje na řízení pohybu ve více osách. Obsahuje bloky jako `MC_CombineAxes` pro synchronizaci více os, `MC_GearIn` a `MC_GearOut` pro převody a `MC_PhasingAbsolute`, `MC_PhasingRelative` pro přesné fázování os. Knihovna nabízí `MC_CamIn` a `MC_CamOut` pro funkce vačky, které umožňují sledování složitých profilů pohybu. `MCP_CamTableSelect` navíc poskytuje flexibilitu při výběru vačkových tabulek a `MC_GearInPos` umožňuje řazení na základě polohy. Tato knihovna je nezbytná pro pokročilé aplikace vyžadující koordinované řízení pohybu napříč více osami.

22 MC_COORD – Koordinované řízení pohybu

Knihovna MC_COORD je speciálně navržena pro koordinaci víceosého řízení pohybu v rámci komplexních systémů. Zahrnuje řadu bloků, včetně `MC_MoveLinearAbsolute` pro provádění přesných lineárních pohybů, doplněných o `MC_MoveLinearRelative` pro relativní lineární pohyb. Pro provádění kruhového pohybu knihovna obsahuje bloky `MC_MoveCircularAbsolute` a `MC_MoveCircularRelative`, což zajišťuje podrobné kruhové trajektorie. V kontextu správy řízení skupinových os tato knihovna zavádí `MC_AddAxisToGroup`, který je dále podporován funkcemi jako `MC_GroupEnable` pro aktivaci, `MC_GroupDisable` pro deaktivaci a `MC_GroupHalt` pro okamžité zastavení seskupených os. Dále knihovna poskytuje `MC_MoveDirectAbsolute` a `MC_MoveDirectRelative`, které umožňují přímou kontrolu nad pohyby os. Pro navigaci po složitých cestách je k dispozici `MC_MovePath`. Základní funkce monitorování a ovládání jsou usnadněny pomocí `MC_GroupReadActualPosition` pro údaje o

poloze, `MC_GroupReadActualVelocity` pro přehled rychlosti, `MC_GroupReadError` pro detekci chyb a `MC_GroupReadStatus` pro aktualizace stavu. Knihovna navíc integruje `MC_ReadCartesianTransform` a `MC_SetCartesianTransform`, které jsou životně důležité pro procesy karteziánské transformace. Tato kolekce funkcí podtrhuje význam knihovny v aplikacích, které vyžadují synchronizované řízení více os, zejména v oblastech robotiky a automatizačních systémů.

23 CanDrv – Komunikace po sběrnici CAN

Knihovna CanDrv je určena pro zpracování komunikace po sběrnici CAN (Controller Area Network) v systému REXYGEN. Obsahuje `CanItem` pro správu datových položek CAN, `CanRecv` pro příjem zpráv ze sběrnice a `CanSend` pro odesílání zpráv. Tato knihovna poskytuje zásadní nástroje pro efektivní a spolehlivou komunikaci přes síť CAN, usnadňuje výměnu dat a řídicích příkazů mezi různými komponentami systému.

24 OpcUaDrv – Komunikace pomocí OPC UA

Knihovna OpcUaDrv se specializuje na propojení se servery OPC UA (Open Platform Communications Unified Architecture) pro průmyslovou automatizaci. První blok – `OpcUaReadValue` je určen pro čtení dat ze serverů, takže je stěžejní pro sběr dat v automatizovaných systémech. Blok `OpcUaWriteValue` umožňuje zápis dat na servery, což umožňuje řízení a provádění příkazů. Blok `OpcUaServerValue` navíc usnadňuje monitorování a správu hodnot serveru. Tato knihovna slouží jako kritický nástroj pro bezproblémovou komunikaci a interakci se servery OPC UA, čímž rozšiřuje možnosti automatizačních systémů.

25 UNIPI – Komunikační bloky pro Unipi

Tato knihovna slouží k ovládání a monitorování zařízení Unipi. Obsahuje bloky pro čtení a zápis digitálních a analogových vstupů a výstupů, bloky pro ovládání relé, PWM výstupů, LED diod a čtení čítačů. Pro čtení sběrnic slouží drivers jako je OwsDrv nebo MbDrv. Bloky fungují v souladu s dokumentací výrobce [1], kde lze najít technické podrobnosti k jednotlivým zařízením a jejich vstupům a výstupům.

Jednotlivé kapitoly příručky na sebe navazují jen volně, a proto mohou být čteny téměř v libovolném pořadí, dokonce může být čtena vždy jen nezbytně nutná informace potřebná k pochopení funkce konkrétního funkčního bloku. Pro tento účel je vhodná zejména elektronická podoba příručky (ve formátu .pdf), vybavená hypertextovými záložkami a obsahem, které usnadňují rychlé nalezení příslušných bloků.

Přesto lze ještě doporučit přečtení následující podkapitoly, která popisuje konvence užívané při popisu bloků ve zbytku příručky.

1.2 Formát popisu funkčních bloků

Popis každého funkčního bloku se skládá z několika sekcí (v uvedeném pořadí):

Symbol bloku – graficky zobrazuje symbolickou značku bloku

Popis funkce – stručně popisuje funkci daného bloku, aniž by byly uváděny příliš detailní informace.

Vstupy – detailně popisuje všechny vstupy daného bloku

Výstupy – detailně popisuje všechny výstupy daného bloku

Parametry – detailně popisuje všechny parametry daného bloku

Příklady – graficky znázorňuje na jednoduchém příkladu použití daného bloku v kontextu ostatních bloků a často uvádí i obrázek s průběhem vstupních a výstupních signálů tak, aby chování bloku bylo přiblíženo co nejnázorněji.

Pokud je funkce bloku zřejmá, nemusí být sekce **Příklady** uvedena. V případě, že blok nemá žádný vstup nebo výstup nebo parametr, není ani příslušná sekce v popisu obsažena.

Vstupy, výstupy a parametry jsou popisovány v tabulkové formě:

<code><jmeno></code>	<code>[jm]</code>	Podrobný popis vstupu (výstupu, parametru) <code><jmeno></code> .	<code><typ></code>
		Matematický symbol <i>jm</i> na pravé straně prvního sloupce je používán ve vzorcích v sekci Popis funkce a bude uváděn, pokud se od jména vstupu liší víc než jen typograficky. Pokud daná proměnná nabývá pouze několika vyjmenovaných hodnot, je význam těchto hodnot uveden v tomto sloupci.	
			<code>[⊙<def>]</code> <code>[↓<min>]</code> <code>[↑<max>]</code>

Význam jednotlivých sloupců je celkem zřejmý. Ve třetím sloupci je vždy uveden pouze `<typ>`. Řídicí systém REXYGEN podporuje typy uvedené v tabulce 1.1. Standardní funkční bloky však nejčastěji používají pro logické proměnné typ `Bool`, pro celočíselné proměnné typ `Long` (I32) a pro reálné proměnné (v pohyblivé řádové čárce) typ `Double` (F64).

Každá takto popsaná proměnná (vstup, výstup či parametr) má v řídicím systému REXYGEN konkrétní implicitní (default) hodnotu `<def>`, uvozenou symbolem $\odot$ a podobně i minimální příp. maximální přípustnou hodnotu, uvozenou symbolem $\downarrow$, příp. $\uparrow$. Všechny tyto tři hodnoty mohou být uvedeny ve druhém sloupci, ale nejsou povinné (jsou umístěny v `[ ]`). Pokud není uvedena hodnota $\odot<def>$, je vždy tato hodnota nulová. Není-li uvedena hodnota $\downarrow<min>$ příp. $\uparrow<max>$, nabývá minimální příp. maximální hodnoty příslušného typu, viz tabulku 1.1¹.

1.3 Konvence pojmenování proměnných, bloků a subsystémů

Pro usnadnění práce s řídicím systémem REXYGEN se používá několik konvencí. V předchozí podkapitole byly zavedeny všechny používané typy proměnných. Pod pojmem pro-

¹Přesný rozsah typu `Large` je -9223372036854775808 až 9223372036854775807.

Typ	Význam	Minimum	Maximum
Bool	logická hodnota 0 nebo 1	0	1
Byte (U8)	8 bit. celé číslo bez znaménka	0	255
Short (I16)	16 bit. celé číslo se znaménkem	-32768	32767
Long (I32)	32 bit. celé číslo se znaménkem	-2147483648	2147483647
Large (I64)	64 bit. celé číslo se znaménkem	$-9.2234 \cdot 10^{18}$	$9.2234 \cdot 10^{18}$
Word (U16)	16 bit. celé číslo bez znaménka	0	65535
DWord (U32)	32 bit. celé číslo bez znaménka	0	4294967295
Float (F32)	32 bit. číslo v pohyblivé ř. čárce	$-3.4 \cdot 10^{38}$	$3.4 \cdot 10^{38}$
Double (F64)	64 bit. číslo v pohyblivé ř. čárce	$-1.7 \cdot 10^{308}$	$1.7 \cdot 10^{308}$
String	znakový řetězec		

Tabulka 1.1: Typy proměnných systému REXYGEN.

měnná budeme mít v této podkapitole na mysli vstupy, výstupy a parametry bloků. Ve velké většině bloků se používají pouze tyto tři typy:

Bool – pro dvouhodnotové logické proměnné, např. zapnuto/vypnuto, ano/ne, pravda/nepravda, true/false, on/off, apod. V této příručce budeme hodnoty logické jedničky (ano, pravda, true, 1) zapisovat jako **on** a hodnoty logické nuly (ne, nepravda, false, 0) jako **off**. To platí i pro vývojové prostředí REXYGEN Studio. V dalších nástrojích a programech třetích stran mohou být jejich hodnoty zobrazovány jako 1 pro **on** a 0 pro **off**. Názvy logických proměnných používají velká písmena, např. **RUN**, **YCN**, **R1**, **UP**.

Long (I32) – pro celočíselné hodnoty, např. číslo sady parametrů, délka trendového bufferu, typ generovaného signálu, chybový kód, výstup čítače, apod. Názvy celočíselných proměnných jsou obvykle psány malými písmeny a počáteční písmeno (vždy malé) je nejčastěji jedno z písmen {i,k,l,m,n,o}, např. **ips**, **l**, **isig**, **iE**, apod. Existuje však několik výjimek z tohoto pravidla, např. **cnt** v bloku **COUNT**, **btype**, **pptype1**, **pfac** a **afac** v bloku **TRND**, apod.

Double (F64) – pro čísla v pohyblivé řádové čárce (reálná), např. zesílení, saturační meze, výsledky většiny matematických funkcí, parametry PID regulátorů, délky časových intervalů v sekundách, apod. Názvy proměnných v pohyblivé řádové čárce používají pouze malá písmena, např. **k**, **hilim**, **y**, **ti**, **tt**.

Typy funkčních bloků v řídicím systému jsou pojmenovávány velkými písmeny, uvnitř jména se mohou vyskytovat číslice a znak '_' (podtržítko). Při vytváření uživatelských instancí bloků doporučujeme na začátku ponechat název typu bloku a doplnit jej o uživatelský název, kde doporučujeme používat všechny uvedené typy znaků a navíc malá písmena.

Výslovně se nedoporučuje používat v uživatelských názvech bloků a vytvořených subsystémů znaky s diakritikou a speciální znaky jako jsou mezery, znaky konce řádků, interpunkční znaménka, operátory, apod. Použití těchto znaků omezuje přenositelnost vytvořených algoritmů na různé platformy a může vést k velké nesrozumitelnosti. Jména

jsou kontrolována překladačem REXYGEN Compiler a pokud obsahují některý z nevhodných znaků je hlášeno varování.

1.4 Kvalita signálu používaná v OPC

Každý signál (vstup, výstup, parametr) v řídicím systému REXYGEN má kromě své hodnoty některého z typů uvedených v tab. 1.1 ještě tzv. *příznaky kvality*. Příznaky kvality používané v řídicím systému REXYGEN jsou shodné s příznaky kvality používanými specifikacemi OPC (Open Platform Communications), viz [2] a obsahují jednobajtovou informaci, jejíž struktura je uvedena v tabulce 1.2.

Číslo bitu	7	6	5	4	3	2	1	0
Váha bitu	128	64	32	16	8	4	2	1
Bitová pole	Kvalita		Substatus				Omezení	
	Q	Q	S	S	S	S	L	L
Špatná (BAD)	0	0	S	S	S	S	L	L
Nejistá (UNCERTAIN)	0	1	S	S	S	S	L	L
(Nevyužito v OPC)	1	0	S	S	S	S	L	L
Dobrá (GOOD)	1	1	S	S	S	S	L	L

Tabulka 1.2: Struktura příznaků kvality

Základní druh kvality určují příznaky QQ v nejvyšších dvou bitech. Podle jejich kombinací uvedených v tabulce rozlišujeme kvalitu dobrou (GOOD), nejistou (UNCERTAIN) a špatnou (BAD). Jemnější rozlišení, tzv. substatus poskytují čtyři bity SSSS. Tyto bity mají různý význam pro různou základní kvalitu. Nejnížší dva bity LL informují o tom, zda daná veličina překročila své meze nebo zda má konstantní hodnotu. Podrobnosti a význam ostatních bitů lze nalézt v kap. 6.8 specifikace [2]. Seznam bloků propagujících kvalitu signálu je uveden v tabulce 1.3.

Princip propagace kvality mezi bloky je následující: Vybere se nejhorší kvalita ze všech *datových* vstupů přivedeného na blok a nastaví se na všechny *datové* výstupy. Jakýkoliv nepřipojený vstup má dobrou kvalitu (GOOD). Kvalita na *řídících* vstupech není monitorována, ale *řídící* vstupy mohou ovlivnit propagaci kvality z *datových* vstupů. Na *stavových* výstupech je kvalita vždy dobrá (GOOD).

Například pro blok DEL je vstup u *datový*, protože přenáší data. Vstup R1 je *řídící*, protože slouží k ovládání bloku. Výstup y je *datový*, protože přenáší informaci z bloku dál. Výstup RDY je *stavový*, protože indikuje stav bloku.

Až na výjimky (SAI, VIN, S10F2) kvalita neovlivňuje algoritmus bloku (tj. aktuální hodnotu na výstupech). Některé bloky mohou mít v důsledku svého algoritmu horší kvalitu (UNCERTAIN, BAD) (např. DEL před naplněním vyrovnávací paměti nebo DIV při dělení 0).

ABSR0T	ABS	ADDHEXD	ADD	ANDHEXD	AND	ARLY	ASW
ATMT	AVG	AVSI	AVS	BDHEXD	BITOP	BMHEXD	BPF
CDELSSM	CMP	CNA	CNB	CNDR	CNI	CNR	CNS
CONCAT	COUNT	CSSM	DDELSSM	DELM	DEL	DER	DFIR
DIF	DIV	DSSM	EAS	EATMT	EDGE	EMD	EQ
EVAR	FIND	FIWR	FLCU	FNXY	FNX	FOPDT	FRID
GAIN	GETPA	GETPR	GETPS	GETPX	IADD	IDIV	IMOD
IMUL	INTE	INTSM	IOASYNC	ISSW	ISUB	ITOI	ITOS
KDER	LC	LEN	LIN	LLC	LPBRK	LPF	LPI
MCU	MDLI	MDL	MID	MINMAX	MUL	MVD	NANINF
NOT	NSCL	ORHEXD	OR	OSD	PIDAT	PIDE	PIDGS
PIDMAX	PIDMA	PIDUI	PIDU	POL	POUT	PSMPC	PWM
RDFT	REC	REL	REPLACE	RLIM	RLY	RS	RTOI
RTOS	RTOV	SAT	SC2FA	SCUV	SCU	SELHEXD	SELSOCT
SELU	SETPA	SETPR	SETPS	SETPX	SHIFTOCT	SHLD	SINT
SMHCCA	SMHCC	SOPDT	SPIKE	SQRT	SQR	SR	SSW
STOR	SUB	SWR	SWU	TIMER	TRIM	TSE	UTOI
VDEL	VIN	VTOR	ZV4IS				

Tabulka 1.3: Seznam bloků propagující kvalitu signálu

Kapitola 2

EXEC – Konfigurace exekutivy reálného času

Obsah

ALARMS – Alarmy systému REXYGEN	29
ARC – Archiv systému REXYGEN	32
DATA – Zahrnutí externích souborů	34
EXEC – Exekutiva reálného času	35
HMI – Konfigurace vizualizace	37
INFO – Dodatečné informace o projektu	39
IODRV – Vstupně-výstupní ovladač systému REXYGEN	40
IOTASK – Úloha řídicího systému REXYGEN spouštěná ovladačem	42
LPBRK – Rozpojení zpětné vazby	43
MODULE – Rozšiřující modul systému REXYGEN	44
OPCUA – Konfigurace OPCUA serveru systému REXYGEN	45
OSCALL – Volání funkcí operačního systému	46
PROJECT – Další nastavení projektu	47
QTASK – Rychlá úloha řídicího systému REXYGEN	48
SLEEP – Časovací blok pro Simulink	49
SRTF – Blok pro nastavování příznaků běhu	50
STATELOAD – Načtení stavů a parametrů více bloků	52
STATESAVE – Uložení stavů a parametrů více bloků	54
SYSEVENT – Čtení systémového logu	56
SYSLOG – Zápis do systémového logu	58
TASK – Standardní úloha řídicího systému REXYGEN	59
TIODRV – Vstupně-výstupní ovladač systému REXYGEN s úlohami	62
WWW – Obsah pro interní webserver	64

Knihovna EXEC je nezbytná pro nastavení exekutivy v reálném čase v systému REXYGEN a zahrnuje klíčové bloky jako `EXEC`, `TASK`, `QTASK` a `HMI`. Tyto bloky jsou fundamentální pro správu provádění úkolů, určování priority procesů a pro interakci s rozhraními pro uživatele, čímž podstatně přispívají k efektivitě a ovladatelnosti aplikací v ekosystému REXYGEN.

ALARMS – Alarmy systému REXYGEN

Symbol bloku

Licence: [STANDARD](#)



Popis funkce

Blok se umísťuje do *hlavního souboru projektu* a slouží pro hromadnou definici alarmů. Tyto alarmy se pak aktivují pomocí bloku [ALM](#) nebo [ALMI](#). Alarmy se definují pomocí externího souboru ve formátu `.csv`. Název tohoto souboru je v parametru `afile`. Alarmy lze aktivovat také pomocí bloků [ALB](#), [ALBI](#), [ALN](#), [ALNI](#), ty ale mají všechny parametry ve svém bloku a nepoužívají hromadnou definici pomocí tohoto bloku **ALARMS**.

Soubor má následující sloupce:

id	... Identifikátor alarmu, přes který se odkazuje blok ALM i záznamy v archivu.
level	... Položka level v archivu (pokud je ukládání do archivu povoleno).
archives	... Bitové pole, kde každý bit značí, zda se do archivu s tímto číslem ukládají události spojené s alarmem (jeho začátek, konec, potvrzení). Např. 0=do archivu se neukládá, 1=ukládá se do 1. archivu, 2=ukládá se do 2. archivu, 4=ukládá se do 3. archivu, 3=ukládá se do 1. a 2. archivu, apod.
group	... Rezervováno pro budoucí použití. Jedná se o další hodnotu (nebo bitové pole) kterou lze použít k filtraci alarmů ve vizualizaci.
name	... Název alarmu; může být použit jako identifikátor, proto by měl být jednoznačný.
description	... Obsahuje textový popis alarmu. Do textu je možné vkládat formátovací znaky pro vícejazyčné texty a pro vložení hodnoty spojené s alarmem (associated values), viz níže.

Vícejazyčný popis

REXYGEN podporuje vícejazyčný popis alarmu. Pole popisu musí být ve tvaru:

```
<ID_jazyk1>:<popis jazyk1>|<ID_jazyk2>:<popis jazyk2>|<ID_jazyk3>:<popis jazyk3>
```

Počet jazyků není omezen, ale celková délka textu je maximálně 32765 bajtů. Pokud uživatel nastaví neznámý jazyk, použije se jazyk1 (tj. první možnost podle pořadí zadání). Pokud uživatel nastaví prázdný jazyk (tj. ""), zobrazí se celý text, tj. všechny jazyky včetně formátovacích značek.

Příklad: Pro popis `en:High voltage alarm|cz:Přepětí` uživatel, který si nastaví jazyk "en" uvidí popis alarmu "High voltage alarm", uživatel, který si nastaví jazyk "cz" uvidí

"Přepětí". Ve všech ostatních případech (např. jazyk "de", ale také "eng", "CZ", "en-us", apod.) se zobrazí "High voltage alarm" (ne proto, že by se preferovala angličtina, ale proto, že tato jazyková verze je uvedena první).

Associated values

Do textu (popisu alarmu), kam se má vložit hodnota asociované proměnné, se musí vložit formátovací text. Ten má tvar:

```
%<číslo proměnné>[<formát>][:<počet cifer>[:<přesnost>]],
```

příčemž formát je jeden znak s významem:

b, B ... binární hodnota (zobrazuje se jako **on**, **off**)

d, D ... celočíselná hodnota (zobrazuje se jako dekadické číslo), implicitní formát pro celočíselné typy

x, X ... celočíselná hodnota (zobrazuje se jako hexadecimální číslo)

f, F ... desetinné číslo v obvyklém tvaru, pokud je uvedena přesnost, je to počet desetinných míst

e, E ... desetinné číslo v exponenciálním tvaru, pokud je uvedena přesnost, je to počet desetinných míst

g, G ... desetinné číslo, podle hodnoty se automaticky zvolí formát **F** nebo **E**, implicitní formát pro desetinná čísla

s, S ... textový řetězec

Pokud formát neodpovídá typu, je ignorován a použije se implicitní formát pro daný typ. Pokud je počet znaků ve formátu příliš malý na zobrazení hodnoty, tak se použije více znaků, aby se hodnota zobrazila správně.

Příklady formátů:

%2 ... hodnota 2. proměnné (**av2** u bloku **ALM** nebo **ALMI**)

%1:8:2 ... hodnota 1. proměnné (**av1** u bloku **ALM** nebo **ALMI**) na 2 desetinná místa, doplněná na 8 znaků mezerami zleva

%1e ... hodnota 1. proměnné (**av1** u bloku **ALM** nebo **ALMI**) v exponenciálním tvaru

Bloky **ALB**, **ALBI** associated values nepoužívají. Pro blok **ALN** a **ALNI** jsou čísla associated values přiřazena takto:

1 ... hodnota vstupu **u**

2 ... hodnota parametru **h**

3 ... hodnota parametru **hh**

4 ... hodnota parametru **l**

5 ... hodnota parametru **ll**

6 ... hodnota parametru **tout**

Poznámky:

- Soubor **.csv** může mít jako oddělovač čárku i středník. První řádka s názvy sloupců je nepovinná.

- Řádky v souboru musí být uspořádány podle sloupce `id` vzestupně.
- Při překladu se kontroluje, jestli `id` jsou unikátní a to i proti ostatním blokům zapisujícím do archivu (`TRND`, `ALB`, `ALN`, ...)
- Blok má speciální editor (tlačítko **Configure** v parametrickém dialogu). Toto tlačítko má funkci, že pokud soubor neexistuje, tak se vytvoří s příkladem ve správném formátu. Toto je vhodné využít poprvé pro zjištění správného formátu, i když později je možné soubor vytvářet jinak.
- U bloků `ALB`, `ALBI`, `ALN` a `ALNI` je zavedeno, že pro `lv1 > 127` se alarm považuje za událost, tj. do archivu se ukládá jen začátek (neukládá se konec ani potvrzení). Bloky `ALM` a `ALMI` tuto funkci neimplementují.
- Hodnoty spojené s alarmem se vyhodnocují při vzniku alarmu. Pokud se hodnota během trvání alarmu změní, v alarmovém okně se to nijak neprojeví.
- V HMI komponentě na prohlížení alarmů je ještě sloupec jméno. To je název alarmového bloku (bez typového prefixu, který se často používá), který alarm generuje.

Tento blok nepropaguje kvalitu signálu. Více informací je uvedeno v sekci 1.4.

Parametr

<code>afile</code>	Jméno souboru pro definici alarmů	<code>String</code>

ARC – Archiv systému REXYGEN

Symbol bloku

Licence: [STANDARD](#)



Popis funkce

Blok ARC slouží v systému REXYGEN pro konfiguraci archivů, sloužících pro průběžné zaznamenávání alarmů, událostí a historických trendů přímo na cílovém zařízení. Vstup **prev** prvního z archivů se propojí s výstupem **Archives** bloku [EXEC](#). Další archivy se přidávají propojováním vstupu **prev** s výstupem **next** předchozího archivu. Na každý výstup **next** smí být připojen nejvýše jeden vstup **prev** následujícího archivu, u posledního archivu zůstává výstup **next** nepřipojen. Vzniklá posloupnost určuje pořadí alokace a inicializace jednotlivých archivů v řídicím systému REXYGEN a také určuje index archivu, používaný v parametru **arc** archivačních bloků (viz kap. 10). Archivy jsou číslovány od 1 a jejich maximální počet je omezen na 15 (archiv č. 0 je interní systémový log).

Typ archivu z hlediska zachování dat i po restartu cílového zařízení je určen parametrem **atype**. Přípustné volby závisejí na možnostech cílového zařízení a lze je po úspěšném připojení k danému zařízení zjistit v diagnostice programu REXYGEN Studio. Standardně jsou k dispozici následující možnosti:

- **1: RAM memory:** Archiv je alokován v paměti RAM (po restartu RexCore je nenávratně ztracen).
- **2: Permanent memory:** Archiv je alokován v zálohované paměti, např. CMOS (po restartu RexCore zůstává zachován).
- **3: Disk:** Archiv je ukládán na disk (po restartu RexCore zůstává zachován).

Archivy jsou na cílovém zařízení tvořeny posloupností úložek proměnné délky (optimalizace paměti a disku), z nichž každá obsahuje časovou značku. Proto dalšími parametry archivu jsou celková velikost v bytech **asize** a maximální počet časových značek **nmarks** pro urychlení sekvenčního vyhledávání v archivu.

Frekvenci zápisu hodnot lze ovlivnit parametrem **period**. U zařízení používajících flash paměť nebo SD karty jako disk není vhodné zapisovat hodnoty příliš často, proto je vhodné nastavit tento parametr na hodnotu v řádech minut. Dále je možné vybrat vhodný zdroj časových značek parametrem **timesrc**.

Tento blok nepropaguje kvalitu signálu. Více informací je uvedeno v sekci 1.4.

Vstup

prev

Vstup pro řetězení archivů

Long (I32)

Parametr

atype	Typ archivu	⊙1	Long (I32)
	1 paměť RAM		
	2 zálohovaná paměť		
	3 disk		
asize	Velikost archivu (v bytech)	↓256 ⊙102400	Long (I32)
nmarks	Počet časových značek	↓2 ⊙720	Long (I32)
ldaymax	Maximální velikost archivu za den [byte]	↓1000 ↑2147480000 ⊙1048576	Large (I64)
period	Perioda zapisování dat na disk [s]	⊙60.0	Double (F64)
timesrc	Zdroj časových značek	⊙1	Long (I32)
	1 CORETIMER		
	2 CORETIMER (přesný)		
	3 RTC (UTC)		
	4 RTC (lokální čas)		
	5 PFC		
	6 TSC		

Výstup

next	Výstup pro řetězení archivů	Long (I32)

DATA – Zahrnutí externích souborů

Symbol bloku

Licence: **STANDARD**



Popis funkce

Blok **DATA** je takzvaný "pseudo-blok", který umožňuje vložit na cílové zařízení externí soubory nebo celé adresáře. Jediný soubor, kde lze blok umístit, je hlavní projektový soubor s jediným blokem **EXEC**. Proto je blok zařazen do knihovny **EXEC**.

Cesta k souboru nebo adresáři na vývojovém zařízení je dána parametrem **Source** a může být buď absolutní nebo relativní. Relativní cesta vychází vždy ze složky projektu. Parametr **Target** určuje cílovou složku a název souboru na cílovém zařízení. Cílová složka je vždy uvnitř adresáře **/rex/data** a hodnota parametru **Target** vychází z tohoto adresáře. Pro nahrávání musí být parametr **IncludeData** nastaven na **on**.

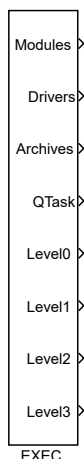
Blok nemá žádné vstupy ani výstupy. Samotný blok **DATA** se nestává součástí konečné binární konfigurace, pouze soubory, na které odkazuje. Buďte opatrní při vkládání velkých souborů nebo adresářů. Data je možné pro přenos zmenšit povolením komprese gzip pomocí parametru **Compression**. Sníží se tím nároky na přenosovou linku, ale mírně se zvýší nároky na procesor cílového zařízení.

Tento blok nepropaguje kvalitu signálu. Více informací je uvedeno v sekci [1.4](#).

EXEC – Exekutiva reálného času

Symbol bloku

Licence: [STANDARD](#)



Popis funkce

Blok **EXEC** tvoří základ tzv. *hlavního souboru projektu* ve formátu `.mdl`, kterým se konfiguruje jednotlivé subsystémy řídicího systému **REXYGEN**, a který nemá analogii v systému Matlab-Simulink. Konfigurace bloku **EXEC** a na něj navázané bloky nerealizují žádný výpočetní algoritmus, ale jsou zpracovány překladačem **REXYGEN Compiler** pro sestavení celé aplikace řídicího systému **REXYGEN**.

Konfigurace systému **REXYGEN** se skládá z modulů (**Modules**), vstupně-výstupních ovladačů (**Drivers**), archivačního subsystému (**Archives**) a subsystému reálného času, obsahujícího rychlou výpočetní úlohu (blíže viz blok **QTASK**) a čtyři prioritní úrovně (**Level0** až **Level3**) pro zařazování výpočetních úloh (blíže viz blok **TASK**).

Parametr **tick** určuje základní (nejkratší) periodu, se kterou bude možno spouštět jednotlivé úlohy. Zadaná hodnota je kontrolována překladačem **REXYGEN Compiler** podle zvoleného cílového zařízení. Obecně lze říci, že čím menší hodnota je zadána, tím je větší režie jádra řídicího systému **REXYGEN**.

Periody jednotlivých výpočetních úrovní **Level0** až **Level3** jsou určeny násobky parametrů **ntick0** až **ntick3** a základní periody **tick**. Parametry **pri0** až **pri3** jsou logickými prioritami odpovídajících výpočetních úrovní v systému **REXYGEN**. Poznamenejme, že systém **REXYGEN** používá 32 logických priorit, kterým jsou interně přiřazeny priority závislé na operačním systému cílového zařízení. Nejvyšší logická priorita systému **REXYGEN** je 0, nejnižší má hodnotu 31, přičemž platí, že pokud mají běžet dvě úlohy s různými prioritami, bude úloha s nižší prioritou (vyšší hodnotou) přerušena úlohou s vyšší prioritou (nižší hodnotou).

Řídicí systém **REXYGEN** vychází z obecně přijímané myšlenky, že „rychlé“ úlohy (s

krátkou periodou vzorkování) je vhodné spouštět s vyšší prioritou než úlohy „pomalé“ (tzv. *Rate monotonic scheduling*). Proto přednastavené hodnoty priorit `pri0` až `pri3` není ve většině případů třeba měnit. Neuvážená změna může vést k těžko předvídatelným důsledkům!

Na zařízeních s více CPU je možné využít parametry `cpu_rt` a `cpu_other`, které umožňují rezervovat jádro pro real-time úlohy (např. `Drivers` a `Levels`) a jádro pro ostatní REXYGEN úlohy (např. diagnostika nebo webový server). Po vyplnění parametru na daném CPU poběží výhradně definované REXYGEN úlohy. Dále lze pomocí parametrů `cpu0` až `cpu3` určit jádro pro daný `Level`. CPU jsou číslovány od 0, přičemž -1 označuje výchozí nastavení. Parametry `cpu0-3` jsou prioritní vůči `cpu_rt` a `cpu_other`.

Tento blok nepropaguje kvalitu signálu. Více informací je uvedeno v sekci 1.4.

Parametr

target		⊙Generic target device	String
<code>tick</code>	Základní perioda runtime jádra [s]	⊙0.05	Double (F64)
<code>ntick0..ntick3</code>	Perioda úloh v úrovni 0 (<code>tick*ntick0</code>)	↓1 ⊙10	Long (I32)
<code>pri0..pri3</code>	Priorita úloh v úrovni 0	↓3 ↑31 ⊙5	Long (I32)
<code>cpu_rt</code>	Výchozí jádro procesoru přiřazené k časově kritickým úlohám (-1=výchozí, 0=jádro 0, 1=jádro 1, ...)	↓-1 ↑127 ⊙-1	Long (I32)
<code>cpu_other</code>	Výchozí jádro procesoru přiřazené k nekritickým úlohám (-1=výchozí, 0=jádro 0, 1=jádro 1, ...)	↓-1 ↑127 ⊙-1	Long (I32)
<code>cpu0..cpu3</code>	Jádro procesoru přiřazené k úrovni 0 (-1=výchozí, 0=jádro 0, 1=jádro 1, ...)	↓-1 ↑127 ⊙-1	Long (I32)

Výstup

<code>Modules</code>	Výstup pro připojení rozšiřujících modulů	Long (I32)
<code>Drivers</code>	Výstup pro připojení I/O ovladačů	Long (I32)
<code>Archives</code>	Výstup pro připojení archivů	Long (I32)
<code>QTask</code>	Výstup pro připojení rychlé úlohy	Long (I32)
<code>Level0..Level3</code>	Výstup pro připojení úloh úrovně 0	Long (I32)

HMI – Konfigurace vizualizace

Symbol bloku

Licence: [STANDARD](#)



Popis funkce

Blok **HMI** je takzvaný "pseudo-blok", který uchovává dodatečná nastavení a parametry související s uživatelským rozhraním (HMI) a obsahem interního webového serveru. Jediný soubor, kde lze blok umístit, je hlavní projektový soubor s jediným blokem [EXEC](#). Proto je blok zařazen do knihovny **EXEC**.

Systém **REXYGEN** aktuálně poskytuje tři jednoduché metody, jak vytvořit uživatelské rozhraní:

- **WebWatch** je automaticky generované HMI z vývojového nástroje **REXYGEN Studio** během kompilace projektu. Má podobný vzhled, atributy a funkce jako online režim vývojového nástroje **REXYGEN Studio**. Hlavním rozdílem je, že **WebWatch** je uložen na cílovém zařízení, je dostupný z integrovaného webového serveru a může být zobrazen v jakémkoliv moderním webovém prohlížeči nebo jakékoli aplikaci kompatibilní s HTML, SVG a JavaScriptem. **WebWatch** je dokonalý nástroj pro okamžité vytvoření HMI, které je vhodné pro vývojáře systémů nebo integrátory. Poskytuje grafickou interakci s téměř všemi signály v řídicím algoritmu.
- **WebBuDi**, což je zkratka pro *Web Buttons and Displays*, je jednoduchý JavaScriptový soubor s několika deklarativními bloky. Tyto bloky popisují datové body, ke kterým je HMI připojeno, a sestavují tabulku, ve které jsou všechna data prezentována. **WebBuDi** poskytuje textovou interakci s vybranými signály a je vhodný pro vývojáře systémů a integrátory nebo může sloužit jako záložní režim HMI pro nestandardní situace.
- **RexHMI** je standardní soubor SVG, který je editován pomocí **REXYGEN HMI Designer**. **REXYGEN HMI Designer** je skvělý nástroj pro vytváření grafického HMI, které je vhodné pro operátory a další koncové uživatele.

Parametr **IncludeHMI** zahrnuje nebo vylučuje soubory HMI z finální binární formy projektu. Parametr **HmiDir** určuje cestu k adresáři, kde se nachází finální HMI a odkud je během kompilace projektu vložen do binárního souboru. Cesta může být absolutní nebo relativní k projektu. Parametr **GenerateWebWatch** určuje, zda má být **WebWatch** HMI generován do **HmiDir** během kompilace. Parametr **GenerateRexHMI** určuje, zda má být **RexHMI** a **WebBuDi** generován do **HmiDir** během kompilace.

Logika generování a zahrnutí HMI během kompilace projektu je následující:

1. Smazat veškerý obsah z `HmiDir`, pokud je vybráno `GenerateWebWatch` nebo `GenerateRexHMI`.
2. Generovat RexHMI a WebBuDi z `SourceDir` do `HmiDir`, pokud je povoleno `GenerateRexHMI`. Všechny zdrojové soubory **WebBuDi** by měly být pojmenovány ve formátu `*.hmi.js` a všechny zdrojové soubory **RexHMI** ve formátu `*.hmi.svg`. Vygenerované soubory jsou poté pojmenovány `*.html`.
3. Kopírovat všechny obsahy z `SourceDir` kromě zdrojových souborů **WebBuDi** nebo **RexHMI** do `HmiDir`, pokud je povoleno `IncludeHMI`.
4. Vložit HMI z `HmiDir` do binární konfigurace, pokud je povoleno `IncludeHMI`.

Blok nemá žádné vstupy ani výstupy. Samotný blok **HMI** se nestává součástí konečné binární konfigurace, pouze soubory, na které odkazuje. Buďte opatrní při vkládání velkých souborů nebo adresářů, protože integrovaný webový server není určen pro masivní přenos dat. Data je možné pro přenos zmenšit povolením komprese gzip pomocí parametru `Compression`. Sníží se tím nároky na přenosovou linku, ale mírně se zvýší nároky na procesor cílového zařízení.

Pro správnou funkci bloku **HMI** musí být kompilace spuštěna z vývojového nástroje REXYGEN Studio a musí být nainstalován REXYGEN HMI Designer.

Tento blok nepropaguje kvalitu signálu. Více informací je uvedeno v sekci 1.4.

Parametr

<code>IncludeHMI</code>	Zahrnout soubory HMI do projektu	☐ on	Bool
<code>HmiDir</code>	Výstupní adresář pro soubory vizualizace (HMI)	☐ hmi	String
<code>SourceDir</code>	Zdrojový adresář	☐ hmisrc	String
<code>GenerateWebWatch</code>	Vygenerovat zdrojové soubory pro WebWatch HMI vizualizací z MDL souborů	☐ on	Bool
<code>GenerateRexHMI</code>	Při překladu projektu vygenerovat HMI ze SVG a JS souborů	☐ on	Bool
<code>RedirectToHMI</code>	Webserver bude automaticky přesměrovávat na stránku s HMI	☐ on	Bool
<code>Compression</code>	Aktivovat kompresi dat		Bool

INFO – Dodatečné informace o projektu

Symbol bloku

Licence: [STANDARD](#)



Popis funkce

Blok **INFO** je takzvaný "pseudo-blok", který ukládá textové informace o reálném čase běhu. Jediný soubor, kde lze blok umístit, je hlavní projektový soubor s jediným blokem **EXEC**. Proto je blok zařazen do knihovny **EXEC**.

Blok nemá žádné vstupy ani výstupy. Informace specifikované s tímto blokem se stávají součástí konečné konfigurace, jsou uloženy na cílovém zařízení a mohou být zobrazeny na různých diagnostických obrazovkách, ale nemají žádný vliv na provádění řídicího algoritmu nebo chování cíle.

Tento blok nepropaguje kvalitu signálu. Více informací je uvedeno v sekci [1.4](#).

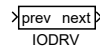
Parametr

Title	Název projektu	String
Author	Autor projektu	String
Description	Stručný popis projektu	String
Customer	Informace o zákazníkovi	String

IODRV – Vstupně-výstupní ovladač systému REXYGEN

Symbol bloku

Licence: **STANDARD**



Popis funkce

Vstupně-výstupní ovladače jsou v systému REXYGEN implementovány jako rozšiřující moduly (viz blok [MODULE](#)). Modul může obsahovat několik ovladačů, které se do konfigurace systému přidávají pomocí bloků IODRV. Vstup **prev** prvního z ovladačů se propojí s výstupem **Drivers** bloku [EXEC](#). Další ovladače se přidávají propojováním vstupu **prev** s výstupem **next** předchozího ovladače. Na každý výstup **next** smí být připojen nejvýše jeden vstup **prev** následujícího ovladače, u posledního ovladače zůstává výstup **next** nepřipojen. Vzniklá posloupnost určuje pořadí inicializace jednotlivých ovladačů do řídicího systému REXYGEN (pořadí zavádění jednotlivých ovladačů je určeno pořadím modulů, v nichž jsou obsaženy, viz popis bloku [MODULE](#)).

Každý ovladač je v systému REXYGEN identifikován svým jménem, které se zadává v parametru **classname**. Pozor, parametr **classname** rozlišuje velká a malá písmena! Pokud se jméno ovladače liší od jména modulu, obsahujícího daný ovladač, musí se zadat i jméno modulu **module**, jinak se ponechá prázdné. Přesné nastavení těchto dvou parametrů je popsáno v příručce pro každý ovladač systému REXYGEN.

Většina ovladačů má svá vlastní konfigurační data uložena v souborech s příponou **.rio** (REXYGEN Input/Output), jejichž jméno určuje parametr **cfgname**. Soubory **.rio** se vytvářejí na stejném adresáři jako hlavní soubor projektu s příponou **.mdl** v němž je použit tento blok. Konfigurační data ovladačů (např. názvy vstupních/výstupních signálů, jejich připojení na konkrétní fyzické vstupy/výstupy, parametry komunikace se vstupně-výstupním zařízením, apod.) se zadávají ve vestavěných editorech poskytovaných přímo ovladači. V programu REXYGEN Studio systému REXYGEN se editory volají stisknutím tlačítka **Configure** v parametrickém dialogu bloku, v systému Simulink je pro stejnou funkci nutno zaškrtnout pomocné políčko "Tick this checkbox to call IODrv EDIT dialog".

Zbylé parametry bloku určují chování ovladače při běhu řídicího systému REXYGEN a mají význam jen tehdy, pokud ovladač implementuje vlastní úlohu (viz příručku k odpovídajícímu ovladači). Parametr **factor** je násobkem základní periody **tick** bloku [EXEC](#), určujícím periodu spouštění této úlohy (**factor*tick**). Parametr **stack** udává velikost zásobníku v bytech (není-li v příručce k ovladači napsáno jinak, není jej třeba měnit). Parametr **pri** určuje logickou prioritu úlohy ovladače. Nevhodná hodnota priority může kriticky ovlivnit výkonnost celého řídicího systému, proto doporučujeme konzultovat příručku k ovladači a poté si ověřit zatížení řídicího systému (ovladačů, výpočetních úrovní a úloh) v diagnostice programu REXYGEN Studio. Parametr **cpu** lze použít k určení, kde

má vlákno ovladače běžet na zařízeních s více CPU.

Tento blok nepropaguje kvalitu signálu. Více informací je uvedeno v sekci [1.4](#).

Vstup

<code>prev</code>	Vstup pro řetězení I/O ovladačů	Long (I32)

Parametr

<code>module</code>	Jméno modulu		String
<code>classname</code>	Jméno třídy ovladače	<code>⊙DrvClass</code>	String
<code>cfgname</code>	Jméno konfiguračního souboru ovladače	<code>⊙iodrv.rio</code>	String
<code>factor</code>	Faktor spouštění	<code>↓1 ⊙10</code>	Long (I32)
<code>stack</code>	Velikost zásobníku (v bytech)	<code>↓1024 ⊙10240</code>	Long (I32)
<code>pri</code>	Priorita úlohy ovladače	<code>↓1 ↑31 ⊙3</code>	Long (I32)
<code>cpu</code>	Jádro procesoru přiřazené úloze ovladače (-1=standardní)	<code>↓-1 ↑127 ⊙-1</code>	Long (I32)

Výstup

<code>next</code>	Výstup pro řetězení I/O ovladačů	Long (I32)

IOTASK – Úloha řídicího systému REXYGEN spouštěná ovladačem

Symbol bloku

Licence: [STANDARD](#)



Popis funkce

Standardní úlohy systému REXYGEN jsou do konfigurace zařazovány pomocí bloku [TASK](#) nebo [QTASK](#). Takové úlohy jsou spouštěny systémovým časovačem, jehož tik (`tick`) se konfiguruje v bloku [EXEC](#).

V některých případech však využití systémového časovače nevyhovuje, např. z důvodu příliš dlouhé nejkratší periody spouštění nebo pokud má být úloha spouštěna od externí události (přerušení od vstupního signálu) apod. V takovém případě může úlohu IOTASK spouštět přímo vstupně-výstupní ovladač zkonfigurovaný pomocí bloku [TIODRV](#). Zda je uvedený způsob spouštění úloh v konkrétním ovladači implementován a za jakých podmínek, lze najít v uživatelské příručce daného ovladače.

Poznámka: Parametr MDLOPEN je určen pro interní potřeby systému REXYGEN a není ho možné měnit ručně.

Tento blok nepropaguje kvalitu signálu. Více informací je uvedeno v sekci [1.4](#).

Vstup

<code>prev</code>	Vstup pro řetězení I/O ovladačů	Long (I32)

Parametr

<code>factor</code>	Faktor spouštění	⊙1	Long (I32)
<code>stack</code>	Velikost zásobníku (v bytech)	⊙10240	Long (I32)
<code>filename</code>	Jméno příslušného MDL souboru		String
<code>MDLOPEN</code>	Příznak otevření MDL souboru		Bool

Výstup

<code>next</code>	Výstup pro řetězení I/O úloh	Long (I32)

LPBRK – Rozpojení zpětné vazby

Symbol bloku

Licence: [STANDARD](#)



Popis funkce

Blok LPBRK je pomocným blokem často používaným v řídicích schématech složených z bloků systému REXYGEN. Blok se obvykle umísťuje do všech zpětných vazeb ve schématu. Jeho chování je však v systémech Simulink a REXYGEN odlišné.

V systému Simulink funguje blok LPBRK jako zpoždění signálu o jeden krok. Kdyby nebyl tento blok vložen do každé zpětné vazby, vyhodnotil by systém Simulink (od verze Matlab 6.1), že schéma obsahuje tzv. „rychlou smyčku“ a simulace by po čase selhala.

V systému REXYGEN je při překladu schématu programem REXYGEN Compiler tento blok vypuštěn, avšak ještě před tím způsobí přerušení zpětnovazební smyčky v místě svého výskytu. Pokud po vypuštění všech bloků LPBRK ještě v řídicím schématu zbývá nějaká smyčka, vypíše překladač REXYGEN Compiler varovnou zprávu a zpětnou vazbu rozpojí v místě, které si sám určí. Pro dosažení co nejvyšší kompatibility mezi systémy REXYGEN a Simulink se doporučuje používat blok LPBRK i v konfiguraci řídicího systému REXYGEN.

Poznámka: Od zavedení kvality na výstupech většiny bloků (odvozuje se od kvality na vstupech a stavu bloku) není blok LPBRK z algoritmu vypouštění, ale má na výstupu kvalitu GOOD (signál v pořádku). Původní chování lze vynutit nastavením parametru `RB = on`. Propagace kvality byla zavedena ve verzi 3.0. Hlavní funkce bloku (indikace zpětnovazebního signálu pro určení pořadí vykonávání bloků) zůstává ve všech případech nezměněna.

Tento blok propaguje kvalitu signálu. Více informací je uvedeno v sekci [1.4](#).

Vstup

<code>u</code>	Vstupní signál	Double (F64)

Parametr

<code>RB</code>	Vypustit blok z konfigurace (jen označení zpětné vazby)	Bool

Výstup

<code>y</code>	Výstupní signál	Double (F64)

MODULE – Rozšiřující modul systému REXYGEN

Symbol bloku

Licence: [STANDARD](#)

Popis funkce

Systém REXYGEN má otevřenou architekturu, jeho zabudované funkce lze tedy dále rozšiřovat a doplňovat. Toto rozšiřování je realizováno právě pomocí modulů. Každý modul je určen svým jménem (umístěným pod symbolem bloku). První rozšiřující modul se zařadí do projektu systému REXYGEN tím, že se jeho vstup **prev** propojí s výstupem **Modules** bloku [EXEC](#). Další moduly se přidávají propojováním vstupu **prev** s výstupem **next** předchozího modulu. Na každý výstup **next** smí být připojen nejvýše jeden vstup **prev** následujícího modulu, u posledního modulu zůstává výstup **next** nepřipojen. Vzniklá posloupnost určuje pořadí zavádění jednotlivých modulů a též pořadí jejich inicializace.

Tento blok nepropaguje kvalitu signálu. Více informací je uvedeno v sekci [1.4](#).

Vstup

prev	Vstup pro řetězení modulů	Long (I32)

Výstup

next	Výstup pro řetězení modulů	Long (I32)

OPCUA – Konfigurace OPCUA serveru systému REXYGEN

Symbol bloku

Licence: [STANDARD](#)



Popis funkce

Blok **OPCUA** funguje jako konfigurační nástroj pro OPC UA server a je známý jako "pseudo-blok". Tento blok je možné umístit pouze do hlavního projektového souboru, který obsahuje blok **EXEC**. Z tohoto důvodu je zařazen do kategorie **EXEC**.

Konfigurace serveru se provádí skrze parametr **opcua_cfg**, jehož možnosti konfigurace jsou detailně popsány v dokumentaci [\[3\]](#), viz tabulka 4.2. Parametr výchozí hodnotou obsahuje vzorovou konfiguraci.

Tento blok nepropaguje kvalitu signálu. Více informací je uvedeno v sekci [1.4](#).

Parametr

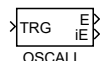
opcua_cfg

String

OSCALL – Volání funkcí operačního systému

Symbol bloku

Licence: [STANDARD](#)



Popis funkce

Blok **OSCALL** je určen pro volání funkcí operačního systému ze systému **REXYGEN**. Zvolená operace je spuštěna vzestupnou hranou (**off**→**on**) na vstupu **TRG**. Na jednotlivých platformách však nemusí být podporovány všechny funkce. Výsledek operace a případný chybový kód jsou indikovány pomocí výstupů **E** a **iE**.

Pro volání externích programů je možno též využít blok [EPC](#).

Tento blok nepropaguje kvalitu signálu. Více informací je uvedeno v sekci [1.4](#).

Vstup

TRG	Spuštění zvolené akce	Bool

Parametr

action	Systémová funkce	⊙1 Long (I32)
	1 restartovat systém	
	2 vypnout systém	
	3 zastavit systém (HALT)	
	4 synchronizace diskových jednotek	
	5 zamknout systémovou partition	
	6 odemknout systémovou partition	
	7 povolit interní webserver	
	8 zakázat interní webserver	
cmd	Rezervováno pro interní použití	String

Výstup

E	Příznak chyby	Bool
	off ... bez chyby	
	on ... nastala chyba	
iE	Kód chyby	Long (I32)

PROJECT – Další nastavení projektu

Symbol bloku

Licence: [STANDARD](#)



Popis funkce

Blok **PROJECT** je takzvaný "pseudo-blok", který ukládá dodatečná nastavení a parametry související s projektem a reálným časem běhu. Jediný soubor, kde lze blok umístit, je hlavní projektový soubor s jediným blokem **EXEC**. Proto patří do kategorie **EXEC**.

Blok nemá žádné vstupy ani výstupy. Informace specifikované s tímto blokem se stávají součástí konečné konfigurace, jsou uloženy na cílovém zařízení a mohou být zobrazeny na různých diagnostických obrazovkách, ale nemají žádný vliv na provádění řídicího algoritmu nebo chování cíle.

Tento blok nepropaguje kvalitu signálu. Více informací je uvedeno v sekci [1.4](#).

Parametr

CompileParams	Parametry příkazového řádku RexComp		String
SourcesOnTarget	Uložit zdrojové soubory na cílové zařízení	☉ on	Bool
TargetURL	Výchozí URL adresa cílového zařízení		String
LibraryPath	Cesta ke knihovnám bloků		String
PreBuild	Příkaz (vykonaný operačním systémem) před překladem		String
PostBuild	Příkaz (vykonaný operačním systémem) po překladu		String

QTASK – Rychlá úloha řídicího systému REXYGEN

Symbol bloku

Licence: [STANDARD](#)



Popis funkce

Blok **QTASK** slouží pro zařazení tzv. rychlé úlohy (quick task) s vysokou prioritou do exekutivy řídicího systému **REXYGEN**. Použití této úlohy je opodstatněné v případech, kdy je nutná co nejrychlejší zpracování vstupních signálů, např. pro číslicovou filtraci vstupních signálů zatížených šumem, nebo pro rychlou odezvu na stisk tlačítek připojených přes logické vstupy. Úloha se zařadí do exekutivy reálného času propojením vstupu **prev** s výstupem **QTask** bloku [EXEC](#). Rychlá úloha se inicializuje před inicializací výpočetní úrovně **Level0** (viz blok [TASK](#)).

Zkonfigurovaná úloha **QTASK** běží s logickou prioritou č. 2 a může být v systému **REXYGEN** nejvýše jedna. Algoritmus této úlohy se konfiguruje stejným způsobem jako algoritmus standardní úlohy **TASK** v samostatném souboru s příponou **.mdl**.

Úloha běží s periodou danou součinem parametru **factor** tohoto bloku a parametru **tick** exekutivy [EXEC](#). Pro hodnotu **factor=1** bude úloha spouštěna s nejkratší periodou **tick** a také zatížení systému bude největší. Pozor, v každé periodě se musí úloha **QTASK** stihnout za dobu kratší než **tick**, v opačném případě dojde k fatální chybě běhu exekutivy reálného času a vykonávání všech úloh se ukončí! Proto by úloha **QTASK** by měla být používána uvážlivě! Naštěstí lze dobu její exekuce zjistit v diagnostice programu **REXYGEN Studio**.

Poznámka: Parametr **MDLOPEN** je určen pro interní potřeby systému **REXYGEN** a není ho možné měnit ručně.

Tento blok nepropaguje kvalitu signálu. Více informací je uvedeno v sekci [1.4](#).

Vstup

prev	Připojení na blok EXEC	Long (I32)

Parametr

factor	Faktor spouštění	⊙1	Long (I32)
stack	Velikost zásobníku (v bytech)	⊙10240	Long (I32)
filename	Jméno příslušného MDL souboru		String
MDLOPEN	Příznak otevření MDL souboru		Bool

SLEEP – Časovací blok pro Simulink

Symbol bloku

Licence: [STANDARD](#)



Popis funkce

Blok **SLEEP** slouží k zajištění co nejpřesnější periody spouštění algoritmu. V řídicím systému REXYGEN je časování výpočetních úloh zajištěno systémovými prostředky (viz blok [EXEC](#)), a proto je blok **SLEEP** ignorován. V systému Matlab/Simulink se pracuje se simulačním časem, který může běžet rychleji nebo pomaleji než reálný čas (podle výkonu počítače a složitosti algoritmu).

Má-li simulace běžet v reálném čase, stačí do simulačního algoritmu zařadit blok **SLEEP**, který jej v každém kroku pozastaví na tak dlouho, aby byl jeho algoritmus volán s periodou danou parametrem **ts**. Mechanismus samozřejmě funguje jen v případě, že simulace běží rychleji než ve skutečnosti.

V současné době je blok **SLEEP** implementován pro systém Matlab/Simulink ve verzi pro operační systémy Windows. Vzhledem k tomu, že ve Windows běží obvykle ještě jiné úlohy, které přerušují simulaci, je vhodné nepoužívat příliš krátké periody v řádu milisekund, doporučená hodnota je od 100 ms. Pro správnou funkci je nutné v parametrech simulace **Solver options** nastavit parametr **Type** na **fixed-step, discrete (no continuous states)** a parametr **Fixed step size** na stejnou hodnotu, jako parametr **ts** bloku **SLEEP**. Blok **SLEEP** by měl být nejvýše jeden v celém simulačním schématu (počítáno včetně subsystémů).

Tento blok nepropaguje kvalitu signálu. Více informací je uvedeno v sekci [1.4](#).

Parametr

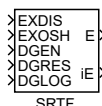
ts Perioda spouštění úlohy [s]

©0.1 Double (F64)

SRTF – Blok pro nastavování příznaků běhu

Symbol bloku

Licence: **ADVANCED**



Popis funkce

Blok **SRTF** (Set Run-Time Flags) slouží pro nastavování příznaků určujících běh úloh, sekvencí (subsystémů) a bloků řídicího systému **REXYGEN**. Tento blok není určen pro Matlab-Simulink. V popisu tohoto bloku bude termín objekt označovat konkrétní objekt řídicího systému **REXYGEN** spouštěný v reálném čase, tj. vstupně-výstupní ovladač, některou z úloh (viz níže), výpočetní sekvenci (subsystém) nebo obyčejný blok systému **REXYGEN**.

Všechny níže uvedené operace jsou prováděny s objektem, jehož úplná cesta je uvedena v parametru **bname**. Není-li tento parametr zadán (prázdný řetězec), provádí se operace s nejbližším vlastníkem daného bloku, tj. pokud je blok obsažen v sekvenci (subsystému) pak s nejbližší nadřazenou sekvencí, jinak přímo s úlohou obsahující daný blok. Úplná cesta k bloku (objektu), rozlišuje malá a velká písmena. Jednotlivé vrstvy jsou oddělovány tečkami, názvy objektů kromě úloh (**TASK**, **QTASK**) začínají jedním z následujících speciálních znaků:

- **^** – výpočetní úroveň (Level), např. **^0** pro **Level0**
- **&** – vstupně-výstupní ovladač (I/O Driver), např. **&WcnDrv**

Jméno úlohy spouštěné vstupně-výstupním ovladačem (**IOTASK**) se zadává ve tvaru **&<jmeno_ovladace>.<jmeno_ulohy>**.

Příznaky bloku umožňují:

- **Zakázat spouštění** daného objektu vstupem **EXDIS** nastaveným na **on**. Spouštění lze opětovně povolit (**EXDIS = off**). Vstup **EXDIS** nastavuje stejný příznak běhu jako tlačítko **Halt/Run** v pravém horním rohu záložky pracovního prostoru bloku (**Workspace**) v diagnostice programu **REXYGEN Studio**.
- **Jednorázově spustit** daný objekt. Pokud je spouštění objektu zakázáno příznakem **EXDIS = on** nebo je zakázáno z diagnostiky v programu **REXYGEN Studio**, lze vstupem **EXOSH = on** (One Shot Execution) spustit daný objekt právě jednou.
- **Povolit zjišťování diagnostických informací** pro objekt vstupem **DGEN = on**. Příznak je shodný s příznakem **Enable** nastavovaným z programu **REXYGEN Studio**

z diagnostických záložek pro jednotlivé objekty (I/O Driver, Level, Quick Task, Task, I/O Task, Sequence).

- **Vynulovat diagnostické informace** pro daný objekt vstupem `DGRES = on`. Příznak je rovněž nastaven z programu **REXYGEN Studio** stisknutím tlačítka **Reset** v diagnostické záložce příslušného objektu. Po vynulování informací je v řídicím systému **REXYGEN** příznak automaticky shozen.

Následující tabulka ukazuje, jaké příznaky lze nastavovat pro různé druhy objektů řídicího systému **REXYGEN**.

Druh objektu	EXDIS	EXOSH	DGEN	DGRES
Vstupně výstupní ovladač (I/O Driver)	✓	✓	✓	✓
Výpočetní úroveň (Level)	✓	×	✓	✓
Výpočetní úloha (Task)	✓	✓	✓	✓
Rychlá úloha (Quick Task)	✓	✓	✓	✓
Úloha vstupně-výstupního ovladače (I/O Task)	✓	✓	✓	✓
Výpočetní sekvence (Sequence, subsystém)	✓	×	✓	✓
Obyčejný blok (Block)	✓	×	×	×

Tento blok nepropaguje kvalitu signálu. Více informací je uvedeno v sekci 1.4.

Vstup

EXDIS	Zakázání spouštění daného objektu	Bool
EXOSH	Jednorázové spuštění daného objektu	Bool
DGEN	Povolení shromažďování diagnostických údajů	Bool
DGRES	Vynulování diagnostických údajů	Bool
DGLOG	Logování rozšířených diagnostických informací	Bool

Parametr

bname	Úplná cesta k řízenému objektu	String

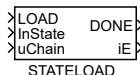
Výstup

E	Příznak chyby	Bool
	off ... bez chyby	
	on nastala chyba	
iE	Kód chyby	Long (I32)
	0 bez chyby	
	1 objekt nenalezen (neplatný parametr bname)	
	2 interní chyba systému REXYGEN (nesprávné ukazatele)	
	3 příznak se nepodařilo nastavit (timeout)	

STATELOAD – Načtení stavů a parametrů více bloků

Symbol bloku

Licence: [ADVANCED](#)



Popis funkce

Blok **STATELOAD** znovu načte hodnoty stavů a parametrů ze souboru nebo řetězce. Soubor je specifikován parametrem **filename** a musí být ve formátu JSON, který je obvykle ukládán blokem [STATESAVE](#). Je také možné načíst data ze vstupu **InState**, který je řetězcem ve formátu JSON stejně jako vstupní soubor. Vstup **InState** se používá, pokud je parametr **filename** prázdný.

Jsou načteny všechny hodnoty, které jsou uloženy v souboru podle konfigurace parametrů **blocks**, **depth** a **mask**. Parametr **blocks** obsahuje relativní cesty (začínající tečkou) k načítaným blokům oddělených středníky. Pokud je **blocks** prázdný, načtou se všechny bloky aktuálního subsystému. Pokud je načítaný blok subsystémem, parametr **depth** určuje, kolik úrovní je také načteno:

- 0: pouze aktuální úroveň,
- n: aktuální úroveň a bloky v subsystémech dalších **n** úrovní.

Dále lze pomocí **mask** určit, které objekty budou načteny. Každý bit čísla znamená:

- 1: vstupy,
- 2: výstupy,
- 4: parametry,
- 8: vnitřní stavy,
- 16: parametry pole,
- 32: stavy pole,
- 64: cyklické (trendové) buffery,
- 256: metadata (pouze **STATESAVE**).

Pokud je parametr **Strict** nastaven na **on**, blok zkontroluje, zda konfigurované bloky a hodnoty odpovídají těm, které jsou uloženy v souboru - pokud není shoda, načtení hodnot se neprovede.

Tento blok nepropaguje kvalitu signálu. Více informací je uvedeno v sekci [1.4](#).

Vstup

<code>LOAD</code>	Načtení stavu	<code>Bool</code>
<code>InState</code>	Řetězec JSON k načtení, pokud je parametr filename prázdný	<code>String</code>
<code>uChain</code>	Užitečný pro umístění bloku ve správném pořadí spuštění	<code>Long (I32)</code>

Parametr

<code>filename</code>	Název souboru pro načtení stavu	<code>String</code>
<code>blocks</code>	Seznam bloků k načtení	<code>String</code>
<code>depth</code>	Určuje, kolik úrovní je také načteno	$\downarrow 0 \uparrow 65535$ <code>Long (I32)</code>
<code>mask</code>	Vyberte, které objekty jsou načteny	$\downarrow 0 \uparrow 65535 \oplus 65535$ <code>Long (I32)</code>
<code>LoadOnInit</code>	Načtení během inicializace konfigurace	$\odot on$ <code>Bool</code>
<code>STRICT</code>	Soubor je zkontrolován vůči aktuální konfiguraci	$\odot on$ <code>Bool</code>

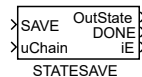
Výstup

<code>DONE</code>	Stav načten	<code>Bool</code>
<code>iE</code>	Kód chyby	<code>Error</code>

STATESAVE – Uložení stavů a parametrů více bloků

Symbol bloku

Licence: [ADVANCED](#)



Popis funkce

Funkční blok **STATESAVE** ukládá hodnoty stavů a parametrů do souboru. Soubor je specifikován parametrem **filename** a je ve formátu JSON, který lze obvykle znovu načíst pomocí bloku **STATELOAD**. Je také možné ukládat data do výstupu **OutState**, který je řetězec JSON ve stejném formátu jako výstupní soubor. Výstup **OutState** se používá, pokud je parametr **filename** prázdný.

Ukládají se všechny hodnoty podle konfigurace parametrů **blocks**, **depth** a **mask**. Parametr **blocks** obsahuje relativní cesty (začínající tečkou) k ukládaným blokům oddělených středníky. Pokud je **blocks** prázdný, uloží se všechny bloky aktuálního subsystému. Pokud je ukládaný blok subsystémem, parametr **depth** určuje, kolik úrovní je také uloženo:

- 0: pouze aktuální úroveň,
- n: aktuální úroveň a bloky v subsystémech dalších **n** úrovní.

Dále lze pomocí **mask** určit, které objekty budou uloženy. Každý bit čísla znamená:

- 1: vstupy,
- 2: výstupy,
- 4: parametry,
- 8: vnitřní stavy,
- 16: parametry pole,
- 32: stavy pole,
- 64: cyklické (trendové) buffery,
- 256: metadata (pouze **STATESAVE**).

Tento blok nepropaguje kvalitu signálu. Více informací je uvedeno v sekci [1.4](#).

Vstup

SAVE	Uložení stavu	Bool
uChain	Užitečný pro umístění bloku ve správném pořadí spuštění	Long (I32)

Parametr

filename	Název souboru, kam se ukládá	String
blocks	Seznam bloků k uložení	String
depth	Určuje, kolik úrovní je uloženo	↓0 ↑65535 Long (I32)
mask	Vyberte, které objekty jsou uloženy	↓0 ↑65535 ⊙65535 Long (I32)
SaveOnExit	Soubor se uloží při ukončení konfigurace	⊙on Bool

Výstup

OutState	Řetězec JSON, kam jsou hodnoty uloženy, pokud je parametr filename prázdný	String
DONE	Stav uložen	Bool
iE	Kód chyby	Error

SYSEVENT – Čtení systémového logu

Symbol bloku

Licence: [STANDARD](#)



Popis funkce

Tento blok slouží ke čtení záznamů ze systémového logu nebo archivu. Čtený archiv se vybere parametrem **arc**. Nezobrazují se všechny položky, ale jen ty co projdou filtrem. Filtrovat lze podle ID položky (u systémového logu nemá význam - momentálně mají všechny úložky **id=1**), podle úrovně alarmu/události (v případě systémového logu jsou tam kódovány kategorie) a v případě textové položky ještě podle hodnoty.

Filtr podle ID se nastavuje pomocí parametrů **idfrom** a **idto**, kterými se zvolí interval, který se zobrazuje. Pokud jsou obě hodnoty stejné, tak se zobrazuje jen jedno **id** a pokud je **idfrom>idto**, tak je filtrování podle **id** vypnuto a zobrazují se všechna **id**.

Filtr podle úrovně se nastavuje pomocí parametrů **lvlfrom** a **lvlto**, přičemž platí stejná pravidla jako v předchozím případě.

Filtr podle hodnoty se uplatňuje jen na textové položky (v systémovém logu jsou to všechny). Položka je zobrazena jen pokud je v ní obsažen text z parametru **filter**. Pokud je parametr prázdný, zobrazují se všechny položky. Na jiné než textové položky nemá tento parametr vliv a vždy se zobrazí (pokud vyhovují nastavení dalších filtrů).

Dokud jsou v archivu položky, které vyhovují filtru, tak se zobrazují tak, že v každém tikku je na výstupu jedna položka (v pořadí, jak jsou uloženy v archivu) a výstup **VALID** je nastaven na hodnotu **on**. Když už není další položka, na výstupech jsou hodnoty odpovídající poslední načtené položce, ale **VALID** je nastaven na **off**.

Výstup **sVal** obsahuje hodnotu textové položky (pro jiné druhy položek je prázdný). Výstup **iVal** obsahuje hodnotu celočíselné položky (pro jiné druhy položek je 0). Ve všech případech jsou všechny parametry (včetně hodnoty) uloženy ve formátu JSON na výstupu **sEvent**. Pro získání potřebných hodnot je možné použít blok **PJSOCT**, popřípadě **PJROCT**.

Poznámka: Pokud se zařadí více bloků **sysevent**, každý prochází příslušný archiv samostatně. Podle nastaveného filtru se pak může stát, že určitá položka z archivu je na výstupu obou bloků, ale obvykle v jiný okamžik.

Tento blok nepropaguje kvalitu signálu. Více informací je uvedeno v sekci 1.4.

Parametr

arc	Číslo čteného archivu (0=systémový logu)	↓0 ↑16	Long (I32)
filter	Text obsažený v položce		String

<code>idfrom</code>	Nejmenší ID položky, které se zobrazuje	↓0 ↑65535	Long (I32)
<code>idto</code>	Největší ID položky, které se zobrazuje	↓0 ↑65535 ⊙65655	Long (I32)
<code>lvlfrom</code>	Nejmenší úroveň položky, které se zobrazuje	↓0 ↑255	Long (I32)
<code>lvlto</code>	Největší ID položky, které se zobrazuje	↓0 ↑255 ⊙255	Long (I32)

Výstup

<code>VALID</code>	Platná (aktuální) výstupní data	Bool
<code>sEvent</code>	Archivní položka (JSON formát)	String
<code>sVal</code>	Hodnota archivní položky (pro text)	String
<code>iVal</code>	Hodnota archivní položky (pro celé číslo)	Long (I32)

SYSLOG – Zápis do systémového logu

Symbol bloku

Licence: [STANDARD](#)



Popis funkce

Blok **SYSLOG** je určený k zapisování libovolných zpráv do systémového logu **REXYGEN**. Lze jej využít na základní logování uživatelských událostí. Pro zápis je potřeba mít v konfiguraci systémového logu povolené zprávy dané úrovně (Target -> System Logs Configuration -> Function block messages).

Tento blok nepropaguje kvalitu signálu. Více informací je uvedeno v sekci [1.4](#).

Vstup

msg	Text zapisovaný do systémového logu	String
lvl	Kategorie zprávy	Long (I32)
	0 Chyba	
	1 Varování	
	2 Informace	
	3 Podrobný	

Parametr

RUN	Zapnutí zápisu do systémového logu	Bool

TASK – Standardní úloha řídicího systému REXYGEN

Symbol bloku

Licence: [STANDARD](#)



Popis funkce

Algoritmy řídicích úloh (task) jsou do systému REXYGEN zařazovány pomocí bloků typu **TASK**. Aplikace řídicího systému může obsahovat několik úloh, které se v konfiguraci systému zařazují do jednotlivých výpočetních úrovní připojením na výstupy **Level0** až **Level3** bloku **EXEC**. Vstup **prev** první úlohy dané úrovně **<i>** se propojí s výstupem **Level<i>** bloku **EXEC**. Další úlohy této úrovně se přidávají propojováním vstupu **prev** s výstupem **next** předchozí úlohy. Na každý výstup **next** smí být připojen nejvýše jeden vstup **prev** následující úlohy stejné úrovně, u poslední úlohy zůstává výstup **next** nepřipojen. Vzniklá posloupnost úloh dané úrovně určuje pořadí inicializace a spouštění úloh této úrovně v řídicím systému REXYGEN. Jednotlivé úrovně se inicializují v pořadí od **Level0** do **Level3** (rychlá úloha **QTASK** se inicializuje před úrovní **Level0**).

Všechny úlohy na dané úrovni **<i>** jsou prováděny se stejnou prioritou, která je určena parametrem **pri<i>** bloku **EXEC**. Doba provádění úlohy je vypočítána jako násobek parametru **factor** a základní periody úrovně **ntick<i>*tick** v bloku **EXEC**.

Čas vyhrazený pro provádění úlohy začíná v tikku **start** a končí v tikku **stop**. Hodnoty **start** a **stop** mohou být pevné nebo automaticky určovány **RexCore**. Pro automatické určování **RexCore** lze parametry vyplnit následovně:

- **start** = -1: Provádění začíná ihned po skončení předchozí úlohy.
- **start** = -2: Provádění začíná v následujícím tikku po dokončení předchozí úlohy.
- **stop** = -1: Provádění úlohy musí skončit před koncem **ntick<i>*tick**.
- **stop** = -2: Provádění úlohy musí skončit v následujícím **tick**.

Pro pevné doby provádění by měly být hodnoty **start** a **stop** nezápornými celými čísly.

Kompilátor **REXYGEN Compiler** dodatečně ověřuje, že hodnota parametru **stop** předchozí úlohy je menší nebo rovna hodnotě parametru **stop** následující úlohy. To zajistí, že přidělené časové intervaly pro jednotlivé úlohy se nepřekrývají. Pokud časování jednotlivých úrovní není vhodné, úlohy mohou být přerušeny úlohami a jinými událostmi s vyšší prioritou. V takových případech provádění není zastaveno, ale pouze zpožděno (na rozdíl od bloku **QTASK**). Sekce **Diagnostics** programu **REXYGEN Studio** hodnotí, zda je zpoždění provádění příležitostné nebo trvalé (karty **Level** a **Task**).

Poznámka: Parametr **MDLOPEN** je určen pro interní potřeby systému REXYGEN a není ho možné měnit ručně.

Tento blok nepropaguje kvalitu signálu. Více informací je uvedeno v sekci [1.4](#).

Vstup

<code>prev</code>	Vstup pro řetězení úloh	Long (I32)

Parametr

<code>factor</code>	Faktor spouštění	⊙1	Long (I32)
<code>start</code>	Startovací tik		Long (I32)
<code>stop</code>	Ukončovací tik	⊙1	Long (I32)
<code>stack</code>	Velikost zásobníku (v bytech)	⊙10240	Long (I32)
<code>filename</code>	Jméno příslušného MDL souboru		String
<code>MDLOPEN</code>	Příznak otevření MDL souboru		Bool

Výstup

<code>next</code>	Výstup pro řetězení úloh	Long (I32)

TIODRV – Vstupně-výstupní ovladač systému REXYGEN s úlohami

Symbol bloku

Licence: [STANDARD](#)



Popis funkce

Blok TIODRV slouží pro konfiguraci speciálních ovladačů řídicího systému REXYGEN, které jsou samy schopny spouštět úlohy konfigurované bloky [IOTASK](#), viz. uživatelská příručka konkrétního ovladače. První z úloh [IOTASK](#) se připojí svým vstupem **prev** na výstup **Tasks** bloku TIODRV. Pokud daný ovladač umožňuje spouštět více než jednu úlohu, připojí se další úloha svým vstupem **prev** na výstup **next** předchozí úlohy [IOTASK](#), atd. Počet připojených úloh a jejich pořadí nekontroluje překladač REXYGEN Compiler (jako v případě bloků [TASK](#)), ale přímo vstupně-výstupní ovladač.

Pokud ovladač nemůže pro některou z úloh zajistit periodické spouštění (např. úloha spouštěná od externí události), nastaví pro tuto úlohu odpovídající příznak. Taková úloha nesmí obsahovat bloky, vyžadující konstantní periodu vzorkování (např. většina regulátorů). V případě, že nějaký ze zakázaných bloků je přesto použit, zahlásí exekutiva chybu běhu úlohy, kterou lze zjistit v diagnostice programu REXYGEN Studio. Parametr **cpu** lze použít k určení, kde má vlákno ovladače běžet na zařízeních s více CPU.

Tento blok nepropaguje kvalitu signálu. Více informací je uvedeno v sekci [1.4](#).

Vstup

prev	Vstup pro řetězení ovladačů (s úlohami)	Long (I32)

Parametr

module	Jméno modulu		String
classname	Jméno třídy ovladače	⊙DrvClass	String
cfname	Jméno konfiguračního souboru ovladače	⊙iodrv.rio	String
factor	Faktor spouštění	↓1 ⊙10	Long (I32)
stack	Velikost zásobníku (v bytech)	↓1024 ⊙10240	Long (I32)
pri	Priorita úlohy ovladače	↓1 ↑31 ⊙3	Long (I32)
cpu	Jádro procesoru přiřazené úloze ovladače (-1=standardní)	↓-1 ↑127 ⊙-1	Long (I32)

Výstup

<code>next</code>	Výstup pro řetězení ovladačů (s úlohami)	Long (I32)
<code>Tasks</code>	Výstup pro připojení I/O úloh	Long (I32)

WWW – Obsah pro interní webserver

Symbol bloku

Licence: [STANDARD](#)



Popis funkce

Blok WWW je takzvaný "pseudo-blok", který ukládá dodatečné informace o obsahu interního webového serveru. Jediný soubor, kde lze blok umístit, je hlavní projektový soubor s jediným blokem EXEC. Proto je blok zařazen do knihovny EXEC.

Blok nemá žádné vstupy ani výstupy. Samotný blok se nestává součástí konečné binární konfigurace, ale data, na která odkazuje, ano. Buďte opatrní při vkládání velkých souborů nebo adresářů, protože integrovaný webový server není optimalizován pro velká data. Je možné data zmenšit povolením komprese gzip. Kompresí se také sníží množství dat přenášovaných klientovi, ale dekompresi musí provést server. Pokud klient nepodporuje kompresi gzip, přináší to další zátěž na cílové zařízení.

Tento blok nepropaguje kvalitu signálu. Více informací je uvedeno v sekci [1.4](#).

Parametr

Source	Zdrojový adresář	String
Target	Cílový adresář	String
Compression	Aktivovat kompresi dat	Bool

Kapitola 3

INOUT – Bloky vstupů a výstupů

Obsah

Display – Zobrazení vstupní hodnoty	67
FromFile – From File	69
FromWorkspace – From Workspace	70
GotoTagVisibility – Viditelnost zdroje signálu	71
INCONN – Blok pro vzdálené získání hodnoty	72
INQUAD, INOCT, INHEXD – Bloky vícenásobných vstupů	73
From, INSTD – Připojení signálu nebo vstupní signál	75
IOASYNC – Asynchronní čtení a zápis	77
OUTCONN – Blok pro nastavení vzdálené hodnoty	78
OUTQUAD, OUTOCT, OUTHEXD – Bloky vícenásobných výstupů	79
OUTRQUAD, OUTROCT, OUTRHEXD – Vícenásobné výstupy s verifikací	82
OUTRSTD – Výstupní signál s verifikací hodnoty	84
Goto, OUTSTD – Zdroj signálu nebo výstupní signál	85
Inport, Outport – Vstupní a výstupní port	87
QFC – Kódování příznaků kvality signálu	89
QFD – Dekódování příznaků kvality signálu	90
SubSystem – Subsystém	91
ToFile – To File	93
ToWorkspace – To Workspace	94
VIN – Ověření kvality vstupního signálu	95
VOUT – Nastavení kvality výstupního signálu	97

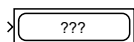
Knihovna INOUT slouží jako klíčové rozhraní v systému REXYGEN, umožňuje plynulou interakci s ovladači vstupů/výstupů. Je navržena pro efektivní simultánní zpracování signálů, což je zásadní pro úlohy rychlého řízení. Tato knihovna zjednodušuje propojení

mezi řídicími algoritmy a hardwarem, zajišťuje minimální latenci. Zároveň poskytuje pokročilé funkce, jako jsou virtuální propojení (vlajky) signálů pro zvýšenou přehlednost schémat a flexibilitu podsystémů.

Display – Zobrazení vstupní hodnoty

Symbol bloku

Licence: [STANDARD](#)



Popis funkce

Blok **Display** zobrazuje vstupní hodnotu ve vybraném formátu. K hodnotě je možné přidat textový sufix specifikováním parametru **suffix**. Aktuální hodnota se okamžitě zobrazí v **REXYGEN Studio** po zapnutí režimu *Watch*. Převod vstupu do jeho textové reprezentace se provádí na cílovém zařízení v každé **Decimation** periodě, takže zobrazenou hodnotu lze také načíst přes rozhraní *REST* nebo použít v uživatelském rozhraní.

Parametr **Format** umožňuje výběr z různých stylů zobrazení hodnoty. Způsoby zobrazení se liší pro neceločíselné a celočíselné hodnoty. Nabízené styly zobrazení zahrnují:

Pro neceločíselné hodnoty:

- **1: Best fit**: Výchozí zobrazení většiny čísel ve stylu "Full Precision", pro extrémně malé nebo velké hodnoty přechází na styl "Scientific Long".
- **2: short**: Zobrazuje hodnoty s maximálně 3 desetinnými místy.
- **3: long**: Zobrazuje hodnoty s maximálním počtem desetinných míst (až 15).
- **4: short_e**: Exponenciální (vědecké) zobrazení hodnot s maximálně 3 desetinnými místy.
- **5: long_e**: Exponenciální (vědecké) zobrazení hodnot s maximálním počtem desetinných míst (až 15).
- **6: bank**: Zobrazuje hodnoty s 2 desetinnými místy.

Pro celá čísla:

- **7: hex**: Číslo v hexadecimální soustavě.
- **8: bin**: Číslo ve dvojkové soustavě.
- **9: dec**: Standardní zobrazení čísel v desítkové soustavě.
- **10: oct**: Číslo v osmičkové soustavě.

Tento blok nepropaguje kvalitu signálu. Více informací je uvedeno v sekci [1.4](#).

Vstup

u	Vstupní signál	Any

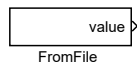
Parametr

Format	Formát zobrazované hodnoty	⊙1	Long (I32)
	short		
	long ..		
	short_e		
	long_e		
	bank ..		
	hex ...		
	bin ...		
	det ...		
Decimation	Po kolika periodách je hodnota zobrazována	↓1 ↑100000 ⊙1	Long (I32)
Suffix	Přípona		String
DispValue	Zobrazená hodnota		String

FromFile – From File

Symbol bloku

Licence: [STANDARD](#)



Popis funkce

Tento blok odpovídá stejnojmennému bloku v Matlabu/Simulinku. Další informace naleznete v dokumentaci Matlab/Simulink.

Tento blok nepropaguje kvalitu signálu. Více informací je uvedeno v sekci [1.4](#).

Parametr

FileName	Název souboru MAT nebo cesta k souboru MAT	String

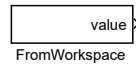
Výstup

value	Výstupní signál	Any

FromWorkspace – From Workspace

Symbol bloku

Licence: [STANDARD](#)



Popis funkce

Tento blok odpovídá stejnojmennému bloku v Matlabu/Simulinku. Další informace naleznete v dokumentaci Matlab/Simulink.

Tento blok nepropaguje kvalitu signálu. Více informací je uvedeno v sekci [1.4](#).

Parametr

VariableName	Jméno nahrávané proměnné	String

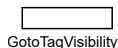
Výstup

value	Výstupní signál	Any

GotoTagVisibility – Viditelnost zdroje signálu

Symbol bloku

Licence: [STANDARD](#)



Popis funkce

Bloky `GotoTagVisibility` upřesňují dostupnost (viditelnost) bloků `Goto` s viditelností `scoped`. Symbol (tag) specifikovaný v bloku `Goto` parametrem `GotoTag` je dostupný ze všech bloků `From` ze subsystému, který obsahuje odpovídající blok `GotoTagVisibility` a též ze všech subsystémů v hierarchii níže.

Blok `GotoTagVisibility` je požadován jen pro takové bloky `Goto`, jejichž parametr `TagVisibility` má hodnotu `scoped`. Pokud má parametr `TagVisibility` hodnoty `local` nebo `global`, není blok `GotoTagVisibility` třeba.

Blok `GotoTagVisibility` se používá jen při překladač projektu překladačem REXY-GEN Compiler a ve výsledné binární konfiguraci není obsažen, protože v reálném čase nevykonává žádnou činnost.

Tento blok nepropaguje kvalitu signálu. Více informací je uvedeno v sekci [1.4](#).

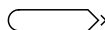
Parametr

<code>GotoTag</code>	Goto odkaz	<code>String</code>

INCONN – Blok pro vzdálené získání hodnoty

Symbol bloku

Licence: [STANDARD](#)



Popis funkce

Blok **INCONN** umožňuje vzdáleně číst hodnoty parametrů (ale i vstupů, a výstupů) jiných bloků, podobně jako bloky [GETPR](#), [GETPS](#), [GETPA](#) a další. Jeho použití je vhodné zejména v situacích vyžadujících rychlou odezvu, například v časově kritických úlohách.

Parametr **sc**, který specifikuje cestu k cílovému parametru, musí být zadán jako *relativní* cesta nebo cesta *relativní k tasku* (viz dokumentace k bloku [GETPR](#)). To znamená, že blok **INCONN** musí být umístěn ve stejné úloze jako cílový parametr, bez ohledu na vzájemnou hierarchickou úroveň. Na rozdíl od bloků **GETP** není možné parametr **sc** upravovat za běhu programu.

Datový typ výstupu **val** je určen typem čtené hodnoty. Oproti blokům **GETP** blok **INCONN** neumožňuje nastavení jednorázového čtení. Hodnota je obnovována v každé periodě spuštění.

Tento blok nepropaguje kvalitu signálu. Více informací je uvedeno v sekci [1.4](#).

Parametr

sc	Jméno vzdáleného parametru	String

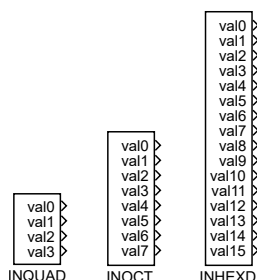
Výstup

val	Hodnota parametru	Any

INQUAD, INOCT, INHEXD – Bloky vícenásobných vstupů

Symboly bloků

Licence: [STANDARD](#)



Popis funkce

Řídicí systém REXYGEN umožňuje kromě čtení každého jednotlivého vstupu z řízeného procesu také současné čtení několika signálů jedním blokem (například vstupů celého modulu nebo zásuvné desky). Pro popsany způsob získávání vstupů slouží bloky INQUAD, INOCT a INHEXD, které se od sebe liší pouze maximálním počtem současně získaných signálů (po řadě 4, 8 a 16).

Název instance bloku zahrnuje symbol ovladače <DRV> a název signálu <signal> daného ovladače:

`<DRV>__<signal>`

Je vytvořen stejným způsobem jako parametr GotoTag bloků INSTD a OUTSTD. Například digitální vstupy zařízení Modbus I/O mohou být odkazovány jako MBM__DI. Podrobné informace o pojmenování signálů lze nalézt v uživatelské příručce příslušného I/O ovladače.

Použití těchto bloků vícenásobných vstupů minimalizuje režii potřebnou k získání signálů prostřednictvím vstupně-výstupních ovladačů, což je významné zejména v případě velmi rychlých řídicích algoritmů s periodou vzorkování do 1 ms a navíc čte všechny uvedené vstupy buď současně nebo po sobě nejrychleji, jak je to možné. Informace, zda je možno pro konkrétní ovladač uvedené bloky používat a jakým způsobem jsou na jejich výstupech vyvedeny vstupy řídicího systému, lze nalézt v uživatelské příručce daného ovladače.

Od verze 2.50.5 je možné použít zástupné symboly v názvech signálů I/O ovladače. To je užitečné uvnitř subsystémů, kde je tento zástupný symbol nahrazen hodnotou parametru subsystému. Např. jméno MBM__modul<id> se bude týkat modulu 1, 2, 3 atd. v závislosti na parametru id subsystému, ve kterém je blok umístěn. Bližší informace o subsystémech a jejich parametrech jsou uvedeny v popisu funkčního bloku [SubSystem](#).

Tento blok nepropaguje kvalitu signálu. Více informací je uvedeno v sekci [1.4](#).

Výstup

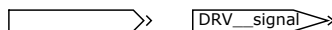
val0..val15 Signál přicházející z I/O ovladače

Any

From, INSTD – Připojení signálu nebo vstupní signál

Symbole bloků

Licence: [STANDARD](#)



Popis funkce

Bloky **From** (připojení signálu) a **INSTD** (standardní vstup) mají stejný symbol a slouží k připojení vstupního signálu do řídicího algoritmu na výstupu **value**. Typ výstupu je dán typem vstupního signálu.

V knihovně bloků naleznete pouze blok **From**. Ten je v případě potřeby při překládání projektu automaticky zkonvertován na blok **INSTD**. O tom, zda daný symbol bloku bude považován za blok **From** nebo **INSTD** rozhoduje překladač **REXYGEN Compiler** podle řetězcového parametru **GotoTag** následovně:

- Obsahuje-li parametr **GotoTag** oddělovač **__** (za sebou dva znaky '_'), jedná se o blok **INSTD**. Část parametru (substring) před tímto oddělovačem (v symbolu bloku výše **DRV**) je považována za jméno bloku typu **IODRV** obsaženého v hlavním souboru projektu. Pokud takový ovladač není v hlavním souboru projektu obsažen, hlásí program **REXYGEN Compiler** chybu. V případě, že takový ovladač v projektu existuje, je druhá část parametru **GotoTag** (za oddělovačem, zde **signal**) považována za jméno vstupního signálu v nalezeném ovladači. Toto jméno je daným ovladačem zkontrolováno a v případě, že ovladač zná vstupní signál s uvedeným jménem, je vytvořena instance bloku **INSTD**, která bude za běhu v reálném čase získávat hodnotu daného vstupního signálu a přivádět ji při každém spuštění dané úlohy do řídicího algoritmu.
- Pokud parametr **GotoTag** oddělovač **__** neobsahuje, je daný blok považován za blok **From**. Při překládání programem **REXYGEN Compiler** se hledá odpovídající blok **Goto** se stejným parametrem **GotoTag** a požadovanou viditelností danou parametrem **TagVisibility** (viz popis bloku **Goto**). V případě, že není nalezen, oznámí překladač **REXYGEN Compiler** varovnou zprávu a blok **From** odstraní. V opačném případě se propojí odpovídající bloky **From** a **Goto**, jako by byly propojeny „neviditelným“ vodičem. Blok **From** se i v tomto případě odstraní a proto nebude obsažen ve výsledné konfiguraci řídicího systému.

V případě bloku **INSTD** obsahuje parametr **GotoTag** symbol ovladače **<DRV>** a název signálu **<signal>** z daného ovladače:

<DRV>__<signal>

Například na první digitální vstup I/O zařízení s komunikací Modbus se může odkazovat pomocí **MBM_DI1**. Detailní informace o pojmenování signálů jsou uvedeny v uživatelské příručce konkrétního I/O ovladače.

Od verze 2.50.5 je možné použít zástupné symboly v názvech signálů I/O ovladače. To je užitečné uvnitř subsystémů, kde je tento zástupný symbol nahrazen hodnotou parametru subsystému. Např. jméno `MBM__DI<id>` se bude týkat vstupu č. 1, 2, 3 atd. v závislosti na parametru `id` subsystému, ve kterém je blok umístěn. Bližší informace o subsystémech a jejich parametrech jsou uvedeny v popisu funkčního bloku [SubSystem](#).

Tento blok nepropaguje kvalitu signálu. Více informací je uvedeno v sekci [1.4](#).

Výstup

<code>value</code>	Signál z I/O ovladače nebo Goto bloku	<code>Any</code>

IOASYNC – Asynchronní čtení a zápis

Symbol bloku

Licence: [STANDARD](#)



Popis funkce

Blok **IOASYNC** je blok pro asynchronní čtení a zápis vstupů a výstupů. V porovnání s bloky [INSTD](#), [OUTSTD](#), [INOCT](#), [OUTOCT](#) a jejich variantami není vykonávání bloku **IOASYNC** řízeno periodickým časováním driveru. Hlavním rozdílem je, že čtení a zápis probíhají výhradně v momentě detekce náběžné hrany **off**→**on** signálu na příslušných vstupech. Zápis hodnoty ze vstupu **u** se aktivuje náběžnou hranou na vstupu **WF**, zatímco čtení do výstupu **y** je vyvoláno náběžnou hranou na vstupu **RF**.

Na rozdíl od ostatních bloků, kde je možné zrušit zápis pomocí nastavení speciálního signálu **_WriteEnable** na **off** (viz dokumentace k danému driveru), blok **IOASYNC** zajišťuje, že při startu exekutivy nedojde ani k jednorázové inicializaci zapisované hodnoty. Tato inicializace může být v některých aplikacích nežádoucí.

Pro navázání spojení s konkrétním signálem driveru je potřeba přejmenovat instanci bloku podle stanoveného formátu názvů signálů driverů, který využívá dvojice podtržítek, podobně jako u bloku [INSTD](#).

Tento blok propaguje kvalitu signálu. Více informací je uvedeno v sekci [1.4](#).

Vstup

u	Signál posílaný do I/O ovladače	Any
RF	Vynutit čtení	Bool
WF	Vynutit zápis	Bool

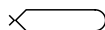
Výstup

y	Signál přicházející z I/O ovladače	Any
BUSY	Příznak probíhající operace	Bool
DONE	Operace dokončena	Bool

OUTCONN – Blok pro nastavení vzdálené hodnoty

Symbol bloku

Licence: [STANDARD](#)



Popis funkce

Blok **OUTCONN** umožňuje vzdáleně nastavovat hodnoty parametrů jiných bloků, podobně jako bloky [SETPR](#), [SETPS](#), [SETPA](#) a další. Jeho použití je vhodné zejména v situacích vyžadujících rychlou odezvu, například v časově kritických úlohách.

Parametr **sc**, který specifikuje cestu k cílovému parametru, musí být zadán jako *relativní* cesta nebo cesta *relativní k tasku* (viz dokumentace k bloku [SETPR](#)). To znamená, že blok **OUTCONN** musí být umístěn ve stejné úloze jako cílový parametr, bez ohledu na vzájemnou hierarchickou úroveň. Pro podrobnosti o formátování cesty k parametru viz dokumentace k bloku [SETPR](#). Na rozdíl od bloků **SETP** není možné parametr **sc** upravovat za běhu programu.

Datový typ vstupu **val** je určen typem připojené hodnoty. Oproti blokům **SETP** blok **OUTCONN** neumožňuje nastavení jednorázového čtení. Hodnota je nastavována v každé periodě spuštění.

Varování: Nastavením připojeného vstupu vzdáleného bloku může dojít k nedefinovanému chování. Stejně tak nastavením parametru bloku uvnitř subsystému, když je tento parametr nastavován v masce subsystému.

Tento blok nepropaguje kvalitu signálu. Více informací je uvedeno v sekci [1.4](#).

Vstup

val	Hodnota parametru	Any

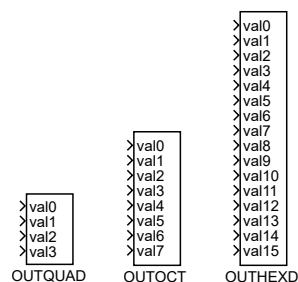
Parametr

sc	Jméno vzdáleného parametru	String

OUTQUAD, OUTOCT, OUTHEXD – Bloky vícenásobných výstupů

Symbole bloků

Licence: [STANDARD](#)



Popis funkce

Řídicí systém REXYGEN umožňuje kromě zápisu každého jednotlivého výstupu z řízeného procesu také současný zápis několika signálů jedním blokem (například výstupů celého modulu nebo zásuvné desky). Pro popsání způsobu nastavování výstupů slouží bloky **OUTQUAD**, **OUTOCT** a **OUTHEXD**, které se od sebe liší pouze maximálním počtem současně zapisovaných signálů (po řadě 4, 8 a 16). Tyto bloky nemají obdobu v knihovně RexLib pro systém Matlab-Simulink.

Název instance bloku zahrnuje symbol ovladače **<DRV>** a název signálu **<signal>** daného ovladače:

<DRV>__<signal>

Je vytvořen stejným způsobem jako parametr **GotoTag** bloků **INSTD** a **OUTSTD**. Například digitální výstupy zařízení Modbus I/O mohou být odkazovány jako **MBM__D0**. Podrobné informace o pojmenování signálů lze nalézt v uživatelské příručce příslušného I/O ovladače.

Použití těchto bloků vícenásobných výstupů minimalizuje režii potřebnou k nastavení signálů prostřednictvím vstupně-výstupních ovladačů, což je významné zejména v případě velmi rychlých řídicích algoritmů s periodou vzorkování do 1 ms a navíc zapisuje všechny uvedené vstupy buď současně nebo po sobě nejrychleji, jak je to možné. Informace, zda je možno pro konkrétní ovladač uvedené bloky používat a jakým způsobem se na jejich vstupy připojují výstupy řídicího systému lze nalézt v uživatelské příručce daného ovladače.

Od verze 2.50.5 je možné použít zástupné symboly v názvech signálů I/O ovladače. To je užitečné uvnitř subsystémů, kde je tento zástupný symbol nahrazen hodnotou parametru subsystému. Např. jméno **MBM__modul<id>** se bude týkat modulu 1, 2, 3 atd. v závislosti na parametru **id** subsystému, ve kterém je blok umístěn. Bližší informace o subsystémech a jejich parametrech jsou uvedeny v popisu funkčního bloku [SubSystem](#).

Tento blok nepropaguje kvalitu signálu. Více informací je uvedeno v sekci [1.4](#).

Vstup

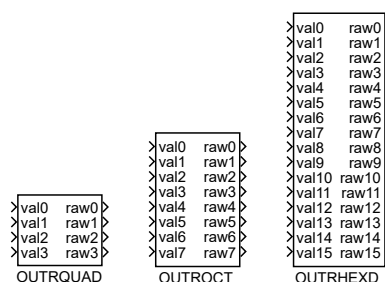
val0..val15 Signál posílaný do I/O ovladače

Any

OUTRQUAD, OUTROCT, OUTRHEXD – Vícenásobné výstupy s verifikací

Symbyoly bloků

Licence: [ADVANCED](#)



Popis funkce

Bloky **OUTRQUAD**, **OUTROCT** a **OUTRHEXD** se používají pro nastavování několika výstupů najednou podobně jako bloky [OUTQUAD](#), [OUTOCT](#) a [OUTHEXD](#). Navíc však umožňují získat pro každý i -tý výstup ovladače přivedený na vstup **val i** zpětnou informaci o výsledku zápisu na odpovídajícím výstupu **raw i** daného bloku.

Výstupy **raw i** mohou být použity k informování řídicího algoritmu o výsledku zápisu dvojím způsobem:

- Hodnotou tohoto výstupu, který může např. u analogového výstupu při překročení maximálního rozsahu A/D převodníku vracet skutečně zapsanou bitovou hodnotu (odtud je v názvu text **raw**).
- Prozkoumáním příznaků kvality tohoto signálu, které lze od signálu oddělit blokem [VIN](#) a dále zpracovat blokem [QFD](#).

Hodnota odpovídající danému zápisu se na výstupech **raw i** nemusí objevit ihned po spuštění daného bloku, ale může mít určité zpoždění dané vlastnostmi použitého ovladače, např. zpožděním komunikace s cílovým zařízením.

Typ a umístění jednotlivých signálů **val** a **raw** jsou popsány v uživatelské příručce příslušného ovladače.

Tento blok nepropaguje kvalitu signálu. Více informací je uvedeno v sekci [1.4](#).

Vstup

val0..val15 Signál posílaný do I/O ovladače

Any

Výstup

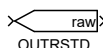
`raw0..raw15` Výsledek nastavení odpovídajícího výstupu

Any

OUTRSTD – Výstupní signál s verifikací hodnoty

Symbol bloku

Licence: [ADVANCED](#)



Popis funkce

Blok **OUTRSTD** se používá pro nastavování výstupu z řídicího algoritmu podobně jako blok [OUTSTD](#). Navíc však umožňuje získat zpětnou informaci o výsledku zápisu na výstupu **raw** daného bloku.

Výstup **raw** může být použit k informování řídicího algoritmu o výsledku zápisu dvojím způsobem:

- Hodnotou tohoto výstupu, která může např. u analogového výstupu při překročení maximálního rozsahu A/D převodníku vracet skutečně zapsanou bitovou hodnotu (odtud je název **raw**).
- Prozkoumáním příznaků kvality tohoto signálu, které lze od signálu oddělit blokem [VIN](#) a dále zpracovat blokem [QFD](#).

Hodnota odpovídající danému zápisu se na výstupu **raw** nemusí objevit ihned po spuštění daného bloku, ale může mít určité zpoždění dané vlastnostmi použitého ovladače, např. zpožděním komunikace s cílovým zařízením.

Tento blok nepropaguje kvalitu signálu. Více informací je uvedeno v sekci [1.4](#).

Vstup

value	Signál posílaný do I/O ovladače	Any

Výstup

raw	Výsledek nastavení odpovídajícího výstupu	Any

Goto, OUTSTD – Zdroj signálu nebo výstupní signál

Symboly bloků

Licence: [STANDARD](#)



Popis funkce

Bloky **Goto** (zdroj signálu) a **OUTSTD** (standardní výstup) mají stejný symbol a slouží k připojení výstupního signálu z řídicího algoritmu.

V knihovně bloků naleznete pouze blok **Goto**. Ten je v případě potřeby při překladu projektu automaticky zkonvertován na blok **OUTSTD**. O tom, zda daný symbol bloku bude považován za blok **Goto** nebo **OUTSTD** rozhoduje překladač **REXYGEN Compiler** podle řetězcového parametru **GotoTag** následovně:

- Obsahuje-li parametr **GotoTag** oddělovač `--` (za sebou dva znaky '_'), jedná se o blok **OUTSTD**. Část parametru (substring) před tímto oddělovačem (v symbolu bloku výše **DRV**) je považována za jméno bloku typu **IODRV** obsaženého v hlavním souboru projektu. Pokud takový ovladač není v hlavním souboru projektu obsažen, hlásí program **REXYGEN Compiler** chybu. V případě, že takový ovladač v projektu existuje, je druhá část parametru **GotoTag** (za oddělovačem, zde **signal**) považována za jméno výstupního signálu v nalezeném ovladači. Toto jméno je daným ovladačem zkontrolováno a v případě, že ovladač zná výstupní signál s uvedeným jménem, je vytvořena instance bloku **OUTSTD**, která bude při každém spuštění dané úlohy v reálném čase nastavovat hodnotu daného výstupního signálu z řídicího algoritmu do ovladače.
- Pokud parametr **GotoTag** oddělovač `--` neobsahuje, je daný blok považován za blok **Goto**. Při překladu programem **REXYGEN Compiler** se hledá odpovídající blok **From** se stejným parametrem **GotoTag**, pro který je tento blok **Goto** viditelný (dosažitelný), viz dále. V případě, že není nalezen, oznámí překladač **REXYGEN Compiler** varovnou zprávu a blok **Goto** odstraní. V opačném případě se propojí odpovídající bloky **Goto** a **From**, jako by byly propojeny „neviditelným“ vodičem. Blok **Goto** se i v tomto případě odstraní a proto nebude obsažen ve výsledné konfiguraci řídicího systému.

Druhý parametr **TagVisibility** bloku **Goto** určuje viditelnost daného bloku uvnitř souboru `.mdl`. Může nabývat těchto hodnot:

- **local:** Oba bloky se musí nacházet ve stejné hierarchické úrovni.
- **global:** Bloky mohou být umístěny kdekoliv v daném `.mdl` souboru.

- **scoped:** Bloky musí být umístěny ve stejném subsystému nebo v jakékoliv hierarchické úrovni pod umístěním bloku `GotoTagVisibility` se stejným parametrem `GotoTag`.

V případě, že je daný blok přeložen jako blok `OUTSTD` je tento parametr ignorován.

V případě bloku `OUTSTD` obsahuje parametr `GotoTag` symbol ovladače `<DRV>` a název signálu `<signal>` z `<DRV>__<signal>`. Například na první digitální výstup I/O zařízení s komunikací Modbus se může odkazovat pomocí `MBM__D01`. Detailní informace o pojmenování signálů jsou uvedeny v uživatelské příručce konkrétního I/O ovladače.

Od verze 2.50.5 je možné použít zástupné symboly v názvech signálů I/O ovladače. To je užitečné uvnitř subsystémů, kde je tento zástupný symbol nahrazen hodnotou parametru subsystému. Např. jméno `MBM__D0<id>` se bude týkat výstupu č. 1, 2, 3 atd. v závislosti na parametru `id` subsystému, ve kterém je blok umístěn. Bližší informace o subsystémech a jejich parametrech jsou uvedeny v popisu funkčního bloku `SubSystem`.

Tento blok nepropaguje kvalitu signálu. Více informací je uvedeno v sekci 1.4.

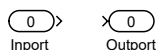
Vstup

<code>value</code>	Signál odesílaný do I/O ovladače nebo bloku <code>From</code>	<code>Any</code>

Inport, Outport – Vstupní a výstupní port

Symbole bloků

Licence: [STANDARD](#)



Popis funkce

Bloky typů vstupní port (**Inport**) a výstupní port (**Outport**) slouží k propojování signálů mezi jednotlivými úrovněmi hierarchie. V systému REXYGEN se používají dvěma způsoby:

1. K připojení vstupů a výstupů subsystému. Bloky realizují přechod mezi symbolickou značkou subsystému a jeho vnitřkem (posloupností bloků skrytých v subsystému). Vlastní značka bloku **Inport** nebo **Outport** je obsažena uvnitř subsystému, jméno daného portu je znázorněno v symbolické značce subsystému v nadřazené hierarchické úrovni.
2. K propojení mezi výpočetními úlohami. V tomto případě jsou bloky obsaženy v nejvyšší hierarchické úrovni dané úlohy (souboru `.mdl`). Příslušné bloky **Inport** a **Outport** musejí mít stejný **Block name**. Propojení mezi bloky v různých úlohách je kontrolováno a vytvořeno kompilátorem REXYGEN Compiler.

V obou případech je pořadí propojovaných vstupních a výstupních signálů určeno parametrem **Port** daného bloku. Číslování vstupních a výstupních portů je navzájem nezávislé, začíná od 1 a v obou případech se provádí v programu REXYGEN Studio automaticky. Čísla portů musí být navíc jednoznačná v dané hierarchické úrovni, a proto v případě ruční změny čísla portu jsou ostatní porty automaticky přechíslovány. Pozor, pokud jsou přechíslovány porty již připojeného subsystému, dojde v důsledku změny pořadí vstupů (nebo výstupů) k změně připojení signálů v nadřazené úrovni subsystému!

V blocích **Inport** a **Outport** je také možné napevno určit datový typ předávané hodnoty pomocí parametru **OutDataTypeStr**. Pokud není vybrána žádná hodnota, resp. je vybrána možnost **Inherit: auto**, je typ hodnoty určen automaticky.

Do parametru **Description** je možné doplnit textový popis bloku. Tento popis je zobrazen ve vlastnostech subsystému a knihovním bloku, pokud je **Inport** nebo **Outport** použit k definování vstupů a výstupů subsystému.

Varování: Tyto bloky nejsou vhodné k propojení polí a jiných odkazů mezi úlohami (výstupy typu odkaz často začínají `ref a` v diagnostice programu REXYGEN Studio mají typ `intptr`). V takovém případě není zajištěno konzistentní čtení a zápis hodnot. V případě pole například mohou být některé hodnoty z jedné periody úlohy a jiné až z další periody. Pro některé struktury může dokonce dojít k pádu RexCore. Pro pole lze využít bloky [SETPA](#) a [GETPA](#). Některé struktury mají toto ošetřené a tam je to výslovně uvedeno v dokumentaci (např. [RM_AxisSpline](#)).

Tento blok nepropaguje kvalitu signálu. Více informací je uvedeno v sekci [1.4](#).

Vstup

<code>value</code>	Hodnota odesílaná na výstupní připojení nebo do bloku Inport	Any

Parametr

<code>OutDataTypeStr</code>	Datový typ hodnoty	String
	<code>Inherit auto</code>	
	<code>double</code>	
	<code>single</code>	
	<code>uint8</code>	
	<code>int16</code>	
	<code>uint16</code>	
	<code>int32</code>	
	<code>uint32</code>	
	<code>boolean</code>	
	<code>float</code>	
	<code>int64</code>	
	<code>string</code>	
<code>Description</code>	Popis portu	String
<code>Port</code>	Pořadí výstupních připojení	Long (I32)

QFC – Kódování příznaků kvality signálu

Symbol bloku

Licence: [STANDARD](#)



Popis funkce

Blok **QFC** vytváří kombinaci tří složek **iq**, **is** a **il** výsledný 8 bitový kód **iqf** příznaků kvality signálu. Příznaky kvality jsou součástí každého vstupního i výstupního signálu v řídicím systému **REXYGEN**. Bližší informace o jejich využití jsou uvedeny v kapitole [1.4](#) této příručky. Knihovna **RexLib** pro Matlab-Simulink příznaky kvality nepoužívá.

Blok **QFC** lze využít v kombinaci s blokem [VOUT](#) pro nastavení potřebných příznaků kvality danému signálu. Obrácenou funkci k bloku **QFC** provádí blok [QFD](#).

Tento blok nepropaguje kvalitu signálu. Více informací je uvedeno v sekci [1.4](#).

Vstup

iq	Základní příznaky kvality	Long (I32)
is	Doplňující příznaky kvality	Long (I32)
il	Příznaky dosažení mezních úrovní	Long (I32)

Výstup

iqf	Bitová kombinace vstupních signálů	Long (I32)

QFD – Dekódování příznaků kvality signálu

Symbol bloku

Licence: [STANDARD](#)



Popis funkce

Blok **QFD** rozkládá 8 bitové příznaky kvality **iqf** na jednotlivé složky **iq**, **is** a **il**. Příznaky kvality jsou součástí každého vstupního i výstupního signálu v řídicím systému REXYGEN. Bližší informace o jejich využití jsou uvedeny v kapitole 1.4 této příručky. Knihovna RexLib pro Matlab-Simulink příznaky kvality nepoužívá.

Blok **QFD** lze využít v kombinaci s blokem [VIN](#) pro detailní zpracování příznaků kvality vstupního signálu u bloku [VIN](#) v řídicím algoritmu. Obrácenou funkci k bloku **QFD** provádí blok [QFC](#).

Tento blok nepropaguje kvalitu signálu. Více informací je uvedeno v sekci 1.4.

Vstup

iqf	Příznaky kvality k dekomponování	Long (I32)

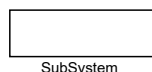
Výstup

iq	Základní příznaky kvality	Long (I32)
is	Doplňující příznaky kvality	Long (I32)
il	Příznaky dosažení mezních úrovní	Long (I32)

SubSystem – Subsystém

Symbol bloku

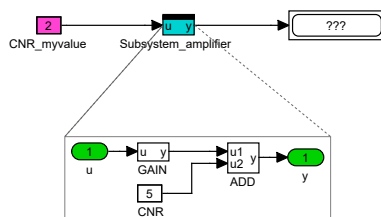
Licence: [STANDARD](#)



Popis funkce

Blok **SubSystem** je základním kamenem hierarchické organizace blokových diagramů v REXYGEN. Subsystém je kontejner pro skupinu funkčních bloků a jejich propojení, které se pak jeví jako jeden blok. Vnořování subsystémů je povoleno, tj. subsystém může zahrnovat další subsystémy.

Při běhu řídicího systému REXYGEN se subsystém vykonává jako seřazená posloupnost bloků, proto je někdy nazýván výpočetní posloupností (anglicky *sequence*). Mezi bloky této posloupnosti není vykonán žádný jiný blok z okolí subsystému. Ty jsou vykonávány buď striktně před nebo striktně po vyhodnocení celého subsystému.



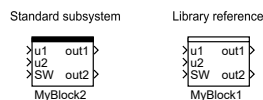
Subsystém může být vytvořen dvěma způsoby:

- Zkopírováním bloku **SubSystem** z knihovny **INOUT** do daného schématu (soubor **.mdl**). Po otevření vytvořeného subsystému mohou být do něj přidávány bloky, včetně vstupních portů **Inport** a výstupních portů **Outport**.
- Označením skupiny bloků a volbou příkazu **Create Subsystem** z menu **Edit**. Vybrané bloky jsou nahrazeny subsystémem, po jehož otevření je možné vidět původní bloky a bloky **Inport** a **Outport**, zprostředkující spojení s bloky v nadřazené (původní) úrovni.

Jakmile je subsystém vytvořen, lze do něj vstoupit pomocí double-kliku.

Pro **SubSystem** je možno vytvořit tzv. masku subsystému a definovat parametry, jejichž hodnoty mohou být využity uvnitř subsystému. Vyberte subsystém a jděte do menu **Edit**→**Declaration of parameters**. Objeví se dialog, ve kterém můžete nadefinovat parametry a jejich popisky (významy). Jakmile je pro subsystém nadefinována maska, začne se chovat jako standardní blok – po double-kliku se objeví dialog *Block properties*. Ten obsahuje parametry definované v masce subsystému. Pokud je potřeba editovat obsah subsystému s maskou, vyberte jej a jděte do menu **Edit**→**Open Subsystem**.

Subsystémy se také používají k vytváření uživatelsky definovaných opakovaně použitelných komponent, které jsou následně umístěny do uživatelských knihoven. Odkaz na knihovnu se liší od standardního subsystému podle stylu horního okraje. Viz obrázek níže.



Podrobnosti o používání subsystémů a vytváření opakovaně použitelných komponent v REXYGEN naleznete v [4].

Použití subsystémů a znovupoužitelných komponent je ilustrováno v příkladech 0101-02 a 0101-03, které je součástí instalace vývojových nástrojů systému REXYGEN.

Tento blok nepropaguje kvalitu signálu. Více informací je uvedeno v sekci 1.4.

ToFile – To File

Symbol bloku

Licence: [STANDARD](#)



Popis funkce

Tento blok odpovídá stejnojmennému bloku v Matlabu/Simulinku. Další informace naleznete v dokumentaci Matlab/Simulink.

Tento blok nepropaguje kvalitu signálu. Více informací je uvedeno v sekci [1.4](#).

Vstup

u	Vstupní signál	Any

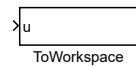
Parametr

FileName	Název souboru MAT nebo cesta k souboru MAT	String

ToWorkspace – To Workspace

Symbol bloku

Licence: [STANDARD](#)



Popis funkce

Tento blok odpovídá stejnojmennému bloku v Matlabu/Simulinku. Další informace naleznete v dokumentaci Matlab/Simulink.

Tento blok nepropaguje kvalitu signálu. Více informací je uvedeno v sekci [1.4](#).

Vstup

u	Vstupní signál	Any

Parametr

VariableName	Jméno proměnné	String

VIN – Ověření kvality vstupního signálu

Symbol bloku

Licence: [STANDARD](#)



Popis funkce

Blok VIN slouží pro ověření kvality vstupního signálu **u** v řídicím systému REXYGEN. Bližší informace o využití příznaků kvality jsou uvedeny v kapitole [1.4](#) této příručky.

Blok průběžně odděluje příznaky kvality vstupu **u** a nastavuje je na výstup **iqf**. Na základě těchto příznaků a parametru **GU** (Good if Uncertain) jsou vstupní signály v bloku VIN dále zpracovány následujícím způsobem:

- Pro **GU = off** je hodnota výstupu **QG** nastavena na **on**, pouze pokud je kvalita vstupu dobrá (GOOD). V případě špatné (BAD) nebo nejisté (UNCERTAIN) kvality je nastaveno **QG = off**.
- Pro **GU = on** je hodnota výstupu **QG** nastavena na **on**, pokud je kvalita vstupu dobrá (GOOD) nebo nejistá (UNCERTAIN). V případě špatné (BAD) kvality je nastaveno **QG = off**.

Je-li vstupní signál **u** vyhodnocen jako kvalitní **QG = on**, je přiveden na výstup **yg**. V případě problémů s kvalitou signálu je pro výstup použit náhradní signál ze vstupu **sv** (substitution variable).

Tento blok propaguje kvalitu signálu. Více informací je uvedeno v sekci [1.4](#).

Vstup

u	Analogový vstupní signál	Any
sv	Náhradní hodnota pro případ chyby	Any

Parametr

GU	Přípustnost kvality UNCERTAIN off ... kvalita UNCERTAIN je nepřípustná on kvalita UNCERTAIN je přípustná	Bool

Výstup

yg	Validní výstupní signál	Any
QG	Indikátor platnosti vstupního signálu	Bool

iqf

Úplné příznaky kvality oddělené od vstupu u

Long (I32)

VOUT – Nastavení kvality výstupního signálu

Symbol bloku

Licence: [STANDARD](#)



Popis funkce

Blok **VOUT** umožňuje signálu **u** nastavit (vnutit) příznaky kvality ze vstupu **iqf**. Bližší informace o využití příznaků kvality jsou uvedeny v kapitole [1.4](#) této příručky.

Tento blok nepropaguje kvalitu signálu. Více informací je uvedeno v sekci [1.4](#).

Vstup

u	Tomuto signálu budou nastaveny příznaky kvality	Any
iqf	Požadované příznaky kvality	Long (I32)

Výstup

yq	Výsledný signál s požadovanými příznaky kvality	Any

Kapitola 4

MATH – Matematické bloky

Obsah

ABS – Absolutní hodnota	101
ADD – Součet dvou signálů	102
ADDQUAD, ADDOCT, ADDHEXD – Součet více signálů	103
CNB – Booleovská (logická) konstanta	104
CNE – Předdefinovaná konstanta	105
CNI – Celočíselná konstanta	106
CNR – Reálná konstanta	107
DIF – Blok difference	108
DIV – Dělení dvou signálů	109
EAS – Rozšířené sčítání a odečítání	110
EMD – Rozšířené násobení a dělení	111
FNX – Výpočet hodnoty funkce jedné proměnné	112
FNXY – Výpočet hodnoty funkce dvou proměnných	114
GAIN – Násobení konstantou	116
GRADS – Gradientní optimalizace	117
IADD – Celočíselné sčítání	119
IDIV – Celočíselné dělení	121
IMOD – Zbytek po celočíselném dělení	122
IMUL – Celočíselné násobení	123
ISUB – Celočíselné odčítání	125
LIN – Lineární interpolace	127
MUL – Násobení dvou signálů	128
NANINF – Blok pro kontrolu hodnot NaN a Inf	129
POL – Vyhodnocení polynomu	131
REC – Převrácená hodnota	132
REL – Relační operace dvou signálů	133

RTOI – Konverze reálného čísla na celé číslo	134
SQR – Druhá mocnina	136
SQRT – Druhá odmocnina	137
SUB – Odčítání dvou signálů	138
UTOI – Konverze celého čísla bez znaménka na celé číslo se znaménkem	139

Knihovna MATH nabízí ucelenou sbírku matematických operací a funkcí. Obsahuje základní aritmetické bloky jako **ADD**, **SUB**, **MUL** a **DIV** pro standardní výpočty a specializovanější bloky, jako je **ABS** pro absolutní hodnoty, **SQRT** pro odmocniny a **SQR** pro odmocnění. Pokročilé funkce poskytují bloky jako **LIN** pro lineární transformace, **POL** pro vyhodnocení polynomů a **FNX**, **FNXY** pro přizpůsobitelné matematické funkce. Knihovna také nabízí operace specifické pro celá čísla prostřednictvím bloků jako **IADD**, **IMUL**, **IDIV** a **IMOD**.

ABS – Absolutní hodnota

Symbol bloku

Licence: [STANDARD](#)



Popis funkce

Blok **ABS** počítá absolutní hodnotu analogového vstupního signálu **u**. Na výstupu **y** je absolutní hodnota vstupu $y = |u|$ a výstup **sgn** určuje znaménko vstupu,

$$\text{sgn} = \begin{cases} -1, & \text{pro } u < 0, \\ 0, & \text{pro } u = 0, \\ 1, & \text{pro } u > 0. \end{cases}$$

Tento blok propaguje kvalitu signálu. Více informací je uvedeno v sekci [1.4](#).

Vstup

u	Analogový vstupní signál	Double (F64)

Výstup

y	Absolutní hodnota vstupního signálu	Double (F64)
sgn	Indikátor znaménka vstupního signálu	Long (I32)

ADD – Součet dvou signálů

Symbol bloku

Licence: [STANDARD](#)



Popis funkce

Blok ADD počítá součet dvou vstupních analogových signálů, výstup je dán vztahem

$$y = u1 + u2.$$

Pro sčítání a odečítání více signálů můžete použít blok [ADD OCT](#).

Tento blok propaguje kvalitu signálu. Více informací je uvedeno v sekci [1.4](#).

Vstup

u1	První analogový vstup bloku	Double (F64)
u2	Druhý analogový vstup bloku	Double (F64)

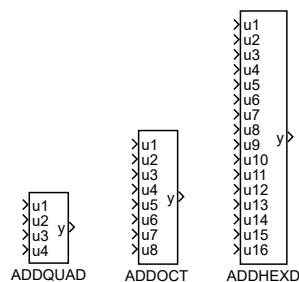
Výstup

y	Součet vstupních signálů	Double (F64)

ADDQUAD, ADDOCT, ADDHEXD – Součet více signálů

Symboly bloků

Licence: [STANDARD](#)



Popis funkce

Bloky ADDQUAD, ADDOCT a ADDHEXD sčítají více (až 16) vstupních signálů. Parametr `n1` udává seznam vstupů, které se místo přičtení odečítají. Pokud je tento parametr prázdný, tak blok provádí funkci $y = u1 + u2 + u3 + u4 + u5 + u6 + u7 + \dots + u16$. Pokud bude například `n1=2,5,7`, tak bude realizována funkce $y = u1 - u2 + u3 + u4 - u5 + u6 - u7 + \dots + u16$.

Pro jednoduché operace sčítání a odečítání můžete použít bloky [ADD](#) a [SUB](#).

Tento blok propaguje kvalitu signálu. Více informací je uvedeno v sekci [1.4](#).

Vstup

<code>u1..u16</code>	Analogový vstupní signál	Double (F64)

Parametr

<code>n1</code>	Seznam signálů k odečtení	Long (I32)

Výstup

<code>y</code>	Součet vstupních signálů	Double (F64)

CNB – Booleovská (logická) konstanta

Symbol bloku

Licence: [STANDARD](#)



Popis funkce

Blok **CNB** slouží pro zadání Booleovské (logické) konstanty.

Tento blok propaguje kvalitu signálu. Více informací je uvedeno v sekci [1.4](#).

Parametr

YCN	Booleovská (logická) konstanta	⊙on Bool
	off ... zakázáno	
	on povoleno	

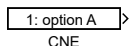
Výstup

Y	Logický výstupní signál	Bool

CNE – Předdefinovaná konstanta

Symbol bloku

Licence: [STANDARD](#)



Popis funkce

Blok **CNE** umožňuje výběr celočíselné konstanty z předem připraveného seznamu. Rozbalovací seznam konstant je definován řetězcem **pupstr**, jehož syntaxe je zřejmá z počáteční hodnoty uvedené níže. Na výstupu bloku je celočíselná hodnota odpovídající číslu ze začátku vybrané položky. V případě, že formát řetězce **pupstr** není správný, je na výstupu bloku 0.

V Simulinku je připravena knihovna CNEs, ve které jsou připraveny bloky **CNE** s nejčastěji používanými seznamy konstant.

Tento blok nepropaguje kvalitu signálu. Více informací je uvedeno v sekci [1.4](#).

Parametr

yenum	Konstanta ze seznamu	⊙1: option A	String
pupstr	Definice seznamu konstant		String
		⊙1: option A 2: option B 3: option C	

Výstup

iy	Celočíselný výstupní signál	Long (I32)

CNI – Celočíselná konstanta

Symbol bloku

Licence: [STANDARD](#)



Popis funkce

Blok CNI slouží pro zadání celočíselné konstanty.

Tento blok propaguje kvalitu signálu. Více informací je uvedeno v sekci [1.4](#).

Parametr

icn	Celočíselná konstanta	⊙1	Long (I32)
vtype	Číselný typ	⊙4	Long (I32)
	2 Byte (U8)		
	3 Short (I16)		
	4 Long (I32)		
	5 Word (U16)		
	6 DWord (U32)		
	10 Large (I64)		

Výstup

iy	Celočíselný výstupní signál	Long (I32)

CNR – Reálná konstanta

Symbol bloku

Licence: [STANDARD](#)



Popis funkce

Blok CNR slouží pro zadání reálné konstanty.

Tento blok propaguje kvalitu signálu. Více informací je uvedeno v sekci [1.4](#).

Parametr

<code>ycn</code>	Reálná konstanta	⊙1.0 Double (F64)

Výstup

<code>y</code>	Analogový výstupní signál	Double (F64)

DIF – Blok difference

Symbol bloku

Licence: [STANDARD](#)



Popis funkce

Blok DIF počítá diferenci vstupního signálu u podle vztahu

$$y_k = u_k - u_{k-1},$$

kde $u = u_k$, $y = y_k$ a u_{k-1} je vstup u zpožděný o jeden krok (o periodu T_S , s níž je blok spouštěn).

Parametr ISSF nastavuje chování bloku v prvním cyklu vykonávání. Pokud je **off**, bude v prvním cyklu na výstupu $y = u$. Pro hodnotu **on** bude v prvním cyklu výstup $y = 0$.

Tento blok propaguje kvalitu signálu. Více informací je uvedeno v sekci [1.4](#).

Vstup

u	Analogový vstupní signál	Double (F64)
R1	Reset bloku	Bool
HLD	Pozastavení	Bool

Parametr

ISSF	Nulový výstup při spuštění	Bool
	off ... nenulový výstup v prvním cyklu	
	on v prvním cyklu je výstup nulový	

Výstup

y	Diference vstupního signálu	Double (F64)

DIV – Dělení dvou signálů

Symbol bloku

Licence: [STANDARD](#)



Popis funkce

Blok **DIV** dělí dva vstupní analogové signály $y = u1/u2$. V případě, že je $u2 = 0$, nastaví se výstup **E** = **on** a na výstup **y** je dána náhradní hodnota $y = yerr$.

Tento blok propaguje kvalitu signálu. Více informací je uvedeno v sekci [1.4](#).

Vstup

u1	První analogový vstup bloku	Double (F64)
u2	Druhý analogový vstup bloku	Double (F64)

Parametr

yerr	Náhradní hodnota pro případ chyby	⊙1.0 Double (F64)

Výstup

y	Podíl vstupních signálů	Double (F64)
E	Indikátor chyby - dělení nulou	Bool
	off ... bez chyby	
	on nastala chyba	

EAS – Rozšířené sčítání a odečítání

Symbol bloku

Licence: [STANDARD](#)



Popis funkce

Blok **EAS** sčítá vstupní analogové signály u_1 , u_2 , u_3 a u_4 s příslušnými vahami a , b , c a d . Výstup y je pak dán vztahem

$$y = a * u_1 + b * u_2 + c * u_3 + d * u_4 + y_0.$$

Tento blok propaguje kvalitu signálu. Více informací je uvedeno v sekci [1.4](#).

Vstup

$u_1 \dots u_4$	Analogový vstupní signál	Double (F64)

Parametr

a	Váhový koeficient pro vstup u_1	⊙1.0	Double (F64)
b	Váhový koeficient pro vstup u_2	⊙1.0	Double (F64)
c	Váhový koeficient pro vstup u_3	⊙1.0	Double (F64)
d	Váhový koeficient pro vstup u_4	⊙1.0	Double (F64)
y_0	Aditivní konstanta (bias)		Double (F64)

Výstup

y	Analogový výstupní signál	Double (F64)

EMD – Rozšířené násobení a dělení

Symbol bloku

Licence: [STANDARD](#)



Popis funkce

Blok **EMD** slouží k násobení a dělení vstupních analogových signálů u_1 , u_2 , u_3 a u_4 s příslušnými vahami a , b , c a d . Výstup y je pak dán vztahem

$$y = \frac{(a * u_1 + a_0)(b * u_2 + b_0)}{(c * u_3 + c_0)(d * u_4 + d_0)}. \quad (4.1)$$

V případě, že jmenovatel vztahu (4.1) je roven 0, nastaví se výstup $E = \text{on}$ a na výstup y je dána náhradní hodnota $y = \text{yerr}$.

Tento blok propaguje kvalitu signálu. Více informací je uvedeno v sekci 1.4.

Vstup

$u_1 \dots u_4$ Analogový vstupní signál Double (F64)

Parametr

a	Váhový koeficient pro vstup u_1	⊙1.0	Double (F64)
a_0	Aditivní konstanta pro vstup u_1		Double (F64)
b	Váhový koeficient pro vstup u_2	⊙1.0	Double (F64)
b_0	Aditivní konstanta pro vstup u_2		Double (F64)
c	Váhový koeficient pro vstup u_3	⊙1.0	Double (F64)
c_0	Aditivní konstanta pro vstup u_3		Double (F64)
d	Váhový koeficient pro vstup u_4	⊙1.0	Double (F64)
d_0	Aditivní konstanta pro vstup u_4		Double (F64)
$yerr$	Náhradní hodnota pro případ chyby	⊙1.0	Double (F64)

Výstup

y	Analogový výstupní signál	Double (F64)
E	Indikátor chyby - dělení nulou	Bool
	off ... bez chyby	
	on nastala chyba	

FNX – Výpočet hodnoty funkce jedné proměnné

Symbol bloku

Licence: [STANDARD](#)



Popis funkce

Blok FNX počítá hodnotu základních matematických funkcí jedné proměnné. Seznam dostupných funkcí s příslušnými omezeními je v níže uvedené tabulce. Vybraná funkce ze seznamu je určena parametrem **ifn**.

Tabulka funkcí bloku FNX:

ifn: zkratka	funkce	omezení u
1: acos	arcus cosinus	$u \in < -1.0, 1.0 >$
2: asin	arcus sinus	$u \in < -1.0, 1.0 >$
3: atan	arcus tangens	–
4: ceil	zaokrouhlení na nejbližší vyšší celé číslo	–
5: cos	cosinus	–
6: cosh	cosinus hyperbolický	–
7: exp	exponenciální křivka e^u	–
8: exp10	exponenciální křivka 10^u	–
9: fabs	absolutní hodnota	–
10: floor	zaokrouhlení na nejbližší nižší celé číslo	–
11: log	logaritmus	$u > 0$
12: log10	dekadický logaritmus	$u > 0$
13: random	náhodné číslo $z < 0, 1 >$ (nezávisí na u)	–
14: sin	sinus	–
15: sinh	sinus hyperbolický	–
16: sqr	druhá mocnina	–
17: sqrt	druhá odmocnina	$u > 0$
18: srand	mění násadu pro funkci random na u	$u \in \mathbb{N}$
19: tan	tangens	–
20: tanh	tangens hyperbolický	–

Poznámka: Všechny trigonometrické funkce pracují s hodnotami v radiánech.

V případě, že vstup u je mimo povolený rozsah nebo nastala chyba při výpočtu funkční hodnoty zvolené funkce (závisí na implementaci), např. výpočet odmocniny záporného čísla, je aktivován chybový výstup **E = on** a na výstup y je nastavena náhradní hodnota $y = \mathbf{yerr}$.

Tento blok propaguje kvalitu signálu. Více informací je uvedeno v sekci [1.4](#).

Vstup

u Analogový vstupní signál **Double (F64)**

Parametr

ifn Typ funkce **⊙1 Long (I32)**

- 1 acos
- 2 asin
- 3 atan
- 4 ceil
- 5 cos
- 6 cosh
- 7 exp
- 8 exp10
- 9 fabs
- 10 floor
- 11 log
- 12 log10
- 13 random
- 14 sin
- 15 sinh
- 16 sqr
- 17 sqrt
- 18 srand
- 19 tan
- 20 tanh

yerr Náhradní hodnota pro případ chyby **Double (F64)**

Výstup

y Výsledek vybrané funkce **Double (F64)**

E Příznak chyby **Bool**

- off ...** bez chyby
- on** nastala chyba

FNXY – Výpočet hodnoty funkce dvou proměnných

Symbol bloku

Licence: [STANDARD](#)



Popis funkce

Blok **FNXY** počítá hodnotu základních matematických funkcí dvou proměnných. Seznam dostupných funkcí s příslušnými omezeními je v níže uvedené tabulce. Vybraná funkce ze seznamu je určena parametrem **ifn**.

Tabulka funkcí bloku **FNXY**:

ifn: zkratka	funkce	omezení $u1, u2$
1: atan2	arcus tangens $u1/u2$	–
2: fmod	zbytek po dělení $u1/u2$	$u2 \neq 0.0$
3: pow	výpočet mocniny $y = u1^{u2}$	viz níže

Funkce **atan2** vrací funkční hodnotu v intervalu $\langle -\pi, \pi \rangle$. Pro určení správného kvadrantu se využívá znamének obou vstupů **u1** a **u2**.

Funkce **fmod** počítá zbytek po dělení $u1/u2$ tak, že platí $u1 = i * u2 + y$, kde i je celé číslo, výstup y má stejné znaménko jako vstup **u1** a pro absolutní hodnotu výstupu y platí: $|y| < |u2|$.

Výpočet mocniny funkcí **pow** se řídí následujícími pravidly:

- Nepracuje se vstupními hodnotami **u1** a **u2** většími než 2^{64} ,
- $u1^0 = 1$ pro libovolné **u1** (i $u1 = 0$),
- 0^{u2} vrací chybu pro $u2 < 0$.

V případě, že vstup **u2** nesplňuje omezení nebo nastala chyba při výpočtu funkční hodnoty zvolené funkce (závisí na implementaci), je aktivován chybový výstup **E = on** a na výstup **y** je nastavena náhradní hodnota **y = yerr**.

Tento blok propaguje kvalitu signálu. Více informací je uvedeno v sekci [1.4](#).

Vstup

u1	První analogový vstup bloku	Double (F64)
u2	Druhý analogový vstup bloku	Double (F64)

Parametr

ifn	Typ funkce	⊙1	Long (I32)
	1 atan2		
	2 fmod		
	3 pow		
yerr	Náhradní hodnota pro případ chyby		Double (F64)

Výstup

y	Výsledek vybrané funkce	Double (F64)
E	Příznak chyby	Bool
	off ... bez chyby	
	on nastala chyba	

GAIN – Násobení konstantou

Symbol bloku

Licence: [STANDARD](#)



Popis funkce

Blok **GAIN** násobí analogový vstup **u** reálnou konstantou **k**. Výstup je pak

$$y = ku.$$

Tento blok propaguje kvalitu signálu. Více informací je uvedeno v sekci [1.4](#).

Vstup

u	Analogový vstupní signál	Double (F64)

Parametr

k	Zesílení	⊙1.0	Double (F64)

Výstup

y	Analogový výstupní signál	Double (F64)

GRADS – Gradientní optimalizace

Symbol bloku

Licence: [ADVANCED](#)



Popis funkce

Blok **GRADS** umožňuje provádět jednodimenzionální minimalizaci funkce $f(\mathbf{x}, v)$ gradientní metodou, kde $\mathbf{x} \in \langle \mathbf{xmin}, \mathbf{xmax} \rangle$ je optimalizační proměnná a y je libovolná vektorová proměnná. Předpokládá se, že pro daný výstup $\mathbf{x}$ v kroku k je hodnota funkce $f(\mathbf{x}, v)$ vyčíslena na vstupu $\mathbf{f}$ v kroku $(k + n)$. To značí, že jednotlivé iterace gradientní metody jsou prováděny s periodou $n * T_S$, kde T_S je perioda spouštění bloku **GRADS**. Délka kroku gradientní metody je určována podle vztahu

$$\begin{aligned} grad &= (\mathbf{f}_i - \mathbf{f}_{i-1}) * (dx)_{i-1} \\ (dx)_i &= -\text{gamma} * grad, \end{aligned}$$

kde k značí číslo iterace. Je-li krok $((dx)_i < \text{dmin})$ nebo $((dx)_i > \text{dmax})$, potom je příslušně omezen.

Tento blok nepropaguje kvalitu signálu. Více informací je uvedeno v sekci [1.4](#).

Vstup

f	Hodnota minimalizované funkce $f(\cdot)$ v bodě $\mathbf{x}$	Double (F64)
x0	Startovní bod optimalizace	Double (F64)
START	Spouštěcí signál (náběžná hrana)	Bool
BRK	Signál pro přerušení	Bool

Parametr

xmin	Dolní mez přípustného intervalu optimální proměnné $\mathbf{x}$	Double (F64)
xmax	Horní mez přípustného intervalu optimální proměnné $\mathbf{x}$	⊙10.0 Double (F64)
gamma	Koeficient gradientní metody určující velikost kroku	⊙0.3 Double (F64)
d0	Počáteční krok gradientní metody	⊙0.05 Double (F64)
dmin	Minimální krok gradientní metody	⊙0.01 Double (F64)
dmax	Maximální krok gradientní metody	⊙1.0 Double (F64)
n	Perioda jedné iterace (v periodách vzorkování bloku)	⊙100 Long (I32)
itermax	Maximální počet iterací před ukončením	⊙20 Long (I32)

Výstup

<code>x</code>	Aktuální hodnota optimalizované proměnné	Double (F64)
<code>xopt</code>	Výsledná optimální hodnota proměnné <code>x</code>	Double (F64)
<code>fopt</code>	Výsledná optimální hodnota funkce <code>f(x)</code>	Double (F64)
<code>BSY</code>	Příznak probíhající operace	Bool
<code>iter</code>	Číslo aktuální iterace	Long (I32)
<code>E</code>	Příznak chyby	Bool
	<code>off ...</code> bez chyby	
	<code>on</code> nastala chyba	
<code>iE</code>	Kód chyby	Long (I32)
	1 <code>x</code> je mimo meze	
	2 <code>x</code> je na mezi	

IADD – Celočíselné sčítání

Symbol bloku

Licence: [STANDARD](#)



Popis funkce

Blok **IADD** sečte dva vstupní celočíselné signály $n = i1 + i2$. V počítači je vždy rozsah celých čísel omezen podle typu proměnné. U tohoto bloku je typ proměnné určen parametrem **vtype**. Pokud se součet vejde do rozsahu proměnné, je výsledkem normální součet. V opačném případě výsledek závisí na hodnotě parametru **SAT**.

Pro **SAT = off** se přetečení rozsahu nekontroluje, tj. nastaví se výstup **E = off** a výstup **n** tak, jak počítá procesor. Například pro typ **Short**, který má rozsah $-32768 \dots +32767$, dostaneme $30000 + 2770 = -32766$.

Pro **SAT = on** se při přetečení rozsahu nastaví výstup **E = on** a na výstup **n** je nejbližší zobrazitelná hodnota (takže pro stejný případ jako výše dostaneme $30000 + 2770 = 32767$).

Tento blok propaguje kvalitu signálu. Více informací je uvedeno v sekci [1.4](#).

Vstup

i1	První celočíselný vstup bloku	$\downarrow -9.22E+18 \uparrow 9.22E+18$	Long (I32)
i2	Druhý celočíselný vstup bloku	$\downarrow -9.22E+18 \uparrow 9.22E+18$	Long (I32)

Parametr

vtype	Číselný typ	⊙4	Long (I32)
	2 Byte (U8)		
	3 Short (I16)		
	4 Long (I32)		
	5 Word (U16)		
	6 DWord (U32)		
	10 Large (I64)		
SAT	Kontrola přetečení		Bool
	off ... přetečení se nekontroluje		
	on ... přetečení se kontroluje		

Výstup

n	Celočíselný součet vstupních signálů	Long (I32)

E	Příznak chyby	Bool
	off ... bez chyby	
	on nastala chyba	

IDIV – Celočíselné dělení

Symbol bloku

Licence: [STANDARD](#)



Popis funkce

Blok IDIV dělí dva vstupní celočíselné signály $n = i1 \div i2$, kde $\div$ označuje operátor celočíselného dělení. Pokud je obyčejný (neceločíselný, normální) podíl obou operandů celé číslo je tato hodnota i výsledkem celočíselného dělení. V opačném případě je výsledkem hodnota, která vznikne „odříznutím“ desetinné části normálního podílu k nejbližšímu celému číslu směrem blíže k nule. V případě, že $i2 = 0$, nastaví se výstup $E = on$ a na výstup n je dána náhradní hodnota $n = nerr$.

Tento blok propaguje kvalitu signálu. Více informací je uvedeno v sekci [1.4](#).

Vstup

$i1$	První celočíselný vstup bloku	$\downarrow -9.22E+18 \uparrow 9.22E+18$	Long (I32)
$i2$	Druhý celočíselný vstup bloku	$\downarrow -9.22E+18 \uparrow 9.22E+18$	Long (I32)

Parametr

$vtype$	Číselný typ	$\odot 4$	Long (I32)
	2 Byte (U8)		
	3 Short (I16)		
	4 Long (I32)		
	5 Word (U16)		
	6 DWord (U32)		
	10 Large (I64)		
$nerr$	Náhradní hodnota pro případ chyby	$\odot 1$	Long (I32)

Výstup

n	Celočíselný podíl vstupních signálů	Long (I32)
E	Indikátor chyby - dělení nulou	Bool
	off ... bez chyby	
	on ... nastala chyba	

IMOD – Zbytek po celočíselném dělení

Symbol bloku

Licence: [STANDARD](#)



Popis funkce

Blok **IMOD** dělí dva vstupní celočíselné signály $n = i1 \% i2$, kde $\%$ označuje operátor zbytku celočíselného dělení (modulo). Pokud jsou obě čísla kladná a dělitel větší než jedna, je výsledek buď nula (pro soudělná čísla) nebo kladné číslo menší než dělitel. V případě, že je jedno z čísel záporné, má výsledek znaménko dělence, např. $15 \% 10 = 5$, $15 \% (-10) = 5$, ale $(-15) \% 10 = -5$. V případě, že $i2 = 0$, nastaví se výstup **E** = **on** a na výstup **n** je dána náhradní hodnota $n = \mathbf{nerr}$.

Tento blok propaguje kvalitu signálu. Více informací je uvedeno v sekci [1.4](#).

Vstup

i1	První celočíselný vstup bloku	↓-9.22E+18 ↑9.22E+18	Long (I32)
i2	Druhý celočíselný vstup bloku	↓-9.22E+18 ↑9.22E+18	Long (I32)

Parametr

vtype	Číselný typ	⊙4	Long (I32)
	2 Byte (U8)		
	3 Short (I16)		
	4 Long (I32)		
	5 Word (U16)		
	6 DWord (U32)		
	10 Large (I64)		
nerr	Náhradní hodnota pro případ chyby	⊙1	Long (I32)

Výstup

n	Zbytek po celočíselném dělení	Long (I32)
E	Indikátor chyby - dělení nulou	Bool
	off ... bez chyby	
	on ... nastala chyba	

IMUL – Celočíselné násobení

Symbol bloku

Licence: [STANDARD](#)



Popis funkce

Blok IMUL vynásobí dva vstupní celočíselné signály $n = i1 * i2$. V počítači je vždy rozsah celých čísel omezen podle typu proměnné. U tohoto bloku je typ proměnné určen parametrem **vtype**. Pokud se součin vejde do rozsahu proměnné, je výsledkem normální součin. V opačném případě výsledek závisí na hodnotě parametru **SAT**.

Pro **SAT = off** se přetečení rozsahu nekontroluje, tj. nastaví se výstup **E = off** a výstup **n** tak jak počítá procesor (například pro typ **Short**, který má rozsah $-32768..+32767$ dostaneme $2000 * 20 = -25536$).

Pro **SAT = on** se při přetečení rozsahu nastaví výstup **E = on** a na výstup **n** je nejbližší zobrazitelná hodnota (takže pro stejný případ jako výše dostaneme $2000 * 20 = 32767$).

Tento blok propaguje kvalitu signálu. Více informací je uvedeno v sekci [1.4](#).

Vstup

i1	První celočíselný vstup bloku	↓-9.22E+18 ↑9.22E+18	Long (I32)
i2	Druhý celočíselný vstup bloku	↓-9.22E+18 ↑9.22E+18	Long (I32)

Parametr

vtype	Číselný typ	⊙4	Long (I32)
	2 Byte (U8)		
	3 Short (I16)		
	4 Long (I32)		
	5 Word (U16)		
	6 DWord (U32)		
	10 Large (I64)		
SAT	Kontrola přetečení		Bool
	off ... přetečení se nekontroluje		
	on ... přetečení se kontroluje		

Výstup

n	Celočíselný součin vstupních signálů	Long (I32)

E	Příznak chyby	Bool
	off ... bez chyby	
	on nastala chyba	

ISUB – Celočíselné odčítání

Symbol bloku

Licence: [STANDARD](#)



Popis funkce

Blok **ISUB** sečte dva vstupní celočíselné signály $n = i1 - i2$. V počítači je vždy rozsah celých čísel omezen podle typu proměnné. U tohoto bloku je typ proměnné určen parametrem **vtype**. Pokud se rozdíl vejde do rozsahu proměnné, je výsledkem normální rozdíl. V opačném případě výsledek závisí na hodnotě parametru **SAT**.

Pro **SAT = off** se přetečení rozsahu nekontroluje, tj. nastaví se výstup **E = off** a výstup **n** tak jak počítá procesor (například pro typ **Short**, který má rozsah $-32768..+32767$ dostaneme $30000 - -2770 = -32766$).

Pro **SAT = on** se při přetečení rozsahu nastaví výstup **E = on** a na výstup **n** je nejbližší zobrazitelná hodnota (takže pro stejný případ jako výše dostaneme $30000 - -2770 = 32767$).

Tento blok propaguje kvalitu signálu. Více informací je uvedeno v sekci [1.4](#).

Vstup

i1	První celočíselný vstup bloku	↓-9.22E+18 ↑9.22E+18	Long (I32)
i2	Druhý celočíselný vstup bloku	↓-9.22E+18 ↑9.22E+18	Long (I32)

Parametr

vtype	Číselný typ	⊙4	Long (I32)
	2 Byte (U8)		
	3 Short (I16)		
	4 Long (I32)		
	5 Word (U16)		
	6 DWord (U32)		
	10 Large (I64)		
SAT	Kontrola přetečení		Bool
	off ... přetečení se nekontroluje		
	on ... přetečení se kontroluje		

Výstup

n	Celočíselný rozdíl vstupních signálů	Long (I32)

E	Příznak chyby	Bool
	off ... bez chyby	
	on nastala chyba	

LIN – Lineární interpolace

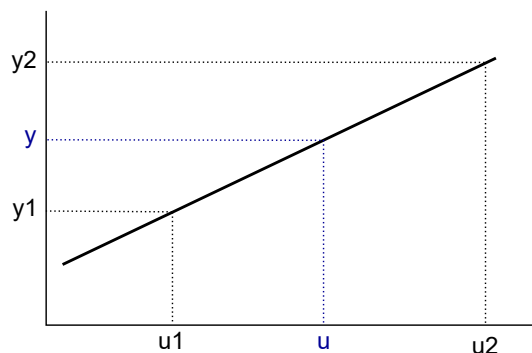
Symbol bloku

Licence: [STANDARD](#)



Popis funkce

Blok LIN počítá lineární interpolaci. Následující obrázek ilustruje výpočet výstupu y ze vstupu u a ze zadaných bodů $[u1, y1]$ a $[u2, y2]$.



Tento blok propaguje kvalitu signálu. Více informací je uvedeno v sekci [1.4](#).

Vstup

u	Analogový vstupní signál	Double (F64)

Parametr

$u1$	Souřadnice prvního bodu v ose x	Double (F64)
$y1$	Souřadnice prvního bodu v ose y	Double (F64)
$u2$	Souřadnice druhého bodu v ose x	⊙1.0 Double (F64)
$y2$	Souřadnice druhého bodu v ose y	⊙1.0 Double (F64)

Výstup

y	Analogový výstupní signál	Double (F64)

MUL – Násobení dvou signálů

Symbol bloku

Licence: [STANDARD](#)



Popis funkce

Blok MUL násobí dva vstupní analogové signály $y = u1 \cdot u2$.

Tento blok propaguje kvalitu signálu. Více informací je uvedeno v sekci [1.4](#).

Vstup

u1	První analogový vstup bloku	Double (F64)
u2	Druhý analogový vstup bloku	Double (F64)

Výstup

y	Součin vstupních signálů	Double (F64)

NANINF – Blok pro kontrolu hodnot NaN a Inf

Symbol bloku

Licence: [STANDARD](#)



Popis funkce

Blok **NANINF** slouží k detekci a korekci neobvyklých hodnot na analogovém vstupu **u**. V případě, že vstupní hodnota je standardní číslo, je přímo přeposílána na výstup **y** ($y = u$) a výstup **is** je nastaven na 0. V situaci, kdy vstupní hodnota nabývá stavu nekonečna ($\pm\text{Inf}$) nebo je typu "není číslo" (not a number - NaN), je na výstup **y** poslána náhradní hodnota definovaná v příslušném parametru **infp**, **infm** nebo **nan** s označením špatné kvality (BAD). Zároveň je na výstup **is** poslán kód označující typ abnormality vstupní hodnoty.

Poznámka: Hodnoty Inf nebo NaN jsou z hlediska matematického koprocessoru téměř normální hodnoty, se kterými umí provádět všechny operace. Vznikají například při dělení nulou nebo při odmocnině ze záporného čísla. Z hlediska řídicího systému je to však nesmyslná hodnota, kterou rozhodně nelze nastavit na fyzický výstup. Bloky systému **REXYGEN** to tak chápou a podobné hodnoty negenerují (např. blok **SQRT** má pro záporná čísla náhradní hodnotu, podobně blok **DIV** pro dělení 0). Někdy se však stane, že podobné nestandardní hodnoty vzniknou a pak je potřeba mít možnost se s tím vypořádat, k čemuž slouží tento blok.

Tento blok propaguje kvalitu signálu. Více informací je uvedeno v sekci [1.4](#).

Vstup

u1	Analogový vstupní signál	Double (F64)

Parametr

infp	Náhradní hodnota pro +Nekonečno	Double (F64)
infm	Náhradní hodnota pro -Nekonečno	Double (F64)
nan	Náhradní hodnota pro Nedefinované číslo	Double (F64)

Výstup

y	Analogový výstupní signál	Double (F64)

<code>is</code>	Status (0:OK, 1:+Inf, 2:-Inf, 3:NaN)	Long (I32)
	0 OK	
	1 +Inf	
	2 -Inf	
	3 NaN	

POL – Vyhodnocení polynomu

Symbol bloku

Licence: [STANDARD](#)



Popis funkce

Blok POL počítá hodnotu polynomiální funkce ve tvaru:

$$y = a_0 + a_1u + a_2u^2 + a_3u^3 + a_4u^4 + a_5u^5 + a_6u^6 + a_7u^7 + a_8u^8.$$

Pro zajištění numerické robustnosti je polynom interně vyhodnocen pomocí Hornerova schématu.

Tento blok propaguje kvalitu signálu. Více informací je uvedeno v sekci [1.4](#).

Vstup

u	Analogový vstupní signál	Double (F64)

Parametr

a0..a8	Koeficient polynomu	Double (F64)

Výstup

y	Analogový výstupní signál	Double (F64)

REC – Převrácená hodnota

Symbol bloku

Licence: [STANDARD](#)



Popis funkce

Blok REC počítá převrácenou hodnotu vstupního signálu u . Výstup je pak

$$y = \frac{1}{u}.$$

Pokud je vstup $u = 0$, nastaví se chybový výstup $E = \text{on}$ a na výstupu y je náhradní hodnota `yerr`.

Tento blok propaguje kvalitu signálu. Více informací je uvedeno v sekci [1.4](#).

Vstup

u	Analogový vstupní signál	Double (F64)

Parametr

<code>yerr</code>	Náhradní hodnota pro případ chyby	$\odot 1.0$ Double (F64)

Výstup

y	Analogový výstupní signál	Double (F64)
E	Indikátor chyby - dělení nulou	Bool
	<code>off</code> ... bez chyby	
	<code>on</code> nastala chyba	

REL – Relační operace dvou signálů

Symbol bloku

Licence: [STANDARD](#)



Popis funkce

Blok REL vyhodnocuje binární relaci $u1 \circ u2$ z hodnot vstupů a podle výsledku relace „ $\circ$ “ je nastavována hodnota výstupu Y na hodnotu **on** (relace platí) nebo **off** (relace neplatí). Kód binární operace je uveden v parametru **irel** popsaném níže.

Tento blok propaguje kvalitu signálu. Více informací je uvedeno v sekci [1.4](#).

Vstup

u1	První analogový vstup bloku	Double (F64)
u2	Druhý analogový vstup bloku	Double (F64)

Parametr

irel	Typ relace	⊙1 Long (I32)
1	 rovnost (==)	
2	 nerovnost (!=)	
3	 menší než (<)	
4	 více než (>)	
5	 menší nebo rovno (<=)	
6	 větší nebo rovno (>=)	

Výstup

Y	Logický výstupní signál	Bool

RTOI – Konverze reálného čísla na celé číslo

Symbol bloku

Licence: [STANDARD](#)



Popis funkce

Blok **RTOI** převádí reálné číslo **r** na celé číslo **i** se znaménkem. Výsledná zaokrouhlená hodnota je určena vztahem:

$$i = \begin{cases} -2147483648, & \text{pro } r \leq -2147483648.0, \\ \text{round}(r), & \text{pro } -2147483648.0 < r \leq 2147483647.0, \\ 2147483647, & \text{pro } r > 2147483647.0, \end{cases}$$

kde $\text{round}(r)$ je zaokrouhlení na nejbližší celé číslo. Čísla ve tvaru $n + 0.5$ (n celé) zaokrouhluje k číslu s vyšší absolutní hodnotou, např. $\text{round}(1.5) = 2$, $\text{round}(-2.5) = -3$.

Poznamenejme, že čísla -2147483648 a 2147483647 odpovídají po řadě nejmenšímu a největšímu číslu se znaménkem zobrazitelným ve formátu s 32 bity (v jazyku C zapsanými v šestnáctkové soustavě jako `0x7FFFFFFF` a `0x80000000`). Tyto limity platí, pokud parametr **vtype** má výchozí hodnotu 4 (long). Jinak platí limity pro příslušný datový typ (viz níže).

Tento blok propaguje kvalitu signálu. Více informací je uvedeno v sekci [1.4](#).

Vstup

r	Analogový vstupní signál	Double (F64)

Parametr

vtype	Číselný typ	⊙4 Long (I32)
	2 Byte (U8)	
	3 Short (I16)	
	4 Long (I32)	
	5 Word (U16)	
	6 DWord (U32)	
	10 Large (I64)	
SAT	Kontrola přetečení	⊙on Bool
	off ... přetečení se nekontroluje	
	on ... přetečení se kontroluje	

Výstup

i

Zaokrouhlený a zkonvertovaný vstupní signál

Long (I32)

SQR – Druhá mocnina

Symbol bloku

Licence: [STANDARD](#)



Popis funkce

Blok **SQR** počítá druhou mocninu analogového vstupu **u**. Výstup je pak

$$y = u^2.$$

Tento blok propaguje kvalitu signálu. Více informací je uvedeno v sekci [1.4](#).

Vstup

u	Analogový vstupní signál	Double (F64)

Výstup

y	Druhá mocnina vstupního signálu	Double (F64)

SQRT – Druhá odmocnina

Symbol bloku

Licence: [STANDARD](#)



Popis funkce

Blok **SQRT** počítá druhou odmocninu analogového vstupu **u**. Výstup je pak

$$y = \sqrt{u}.$$

V případě $u < 0$ je aktivován chybový výstup $E = \text{on}$ a výstup **y** je nastaven na hodnotu parametru **yerr**.

Tento blok propaguje kvalitu signálu. Více informací je uvedeno v sekci [1.4](#).

Vstup

u	Analogový vstupní signál	Double (F64)

Parametr

yerr	Náhradní hodnota pro případ chyby	⊙1.0 Double (F64)

Výstup

y	Odmocnina ze vstupní hodnoty	Double (F64)
E	Příznak chyby	Bool
	off ... bez chyby	
	on nastala chyba	

SUB – Odčítání dvou signálů

Symbol bloku

Licence: [STANDARD](#)



Popis funkce

Blok SUB počítá rozdíl dvou vstupních analogových signálů, výstup je dán vztahem

$$y = u1 - u2.$$

Pro sčítání a odečítání více signálů můžete použít blok [ADDOCT](#).

Tento blok propaguje kvalitu signálu. Více informací je uvedeno v sekci [1.4](#).

Vstup

u1	Analogový vstupní signál	Double (F64)
u2	Analogový vstupní signál	Double (F64)

Výstup

y	Rozdíl mezi vstupními signály	Double (F64)

UTOI – Konverze celého čísla bez znaménka na celé číslo se znaménkem

Symbol bloku

Licence: [STANDARD](#)



Popis funkce

Blok UTOI umožňuje převod neznaménkového celého čísla na znaménkové číslo v reprezentaci dvojkového doplňku, což je běžná reprezentace používaná v procesorech. Například, v 8-bitové reprezentaci je číslo -1 reprezentováno jako 255, a v 16-bitové reprezentaci jako 65535. Rozlišení, kolikabitová reprezentace se má použít, určuje parametr **bits**.

Primární využití tohoto bloku je v situacích, kdy hodnota získaná z ovladače obsahuje více signálů, které se izolují vymaskováním (typicky pomocí bloků [INTSM](#) nebo [BITOP](#)). Výsledkem tohoto vymaskování je vždy neznaménkové (kladné) číslo. Pokud však má být signál interpretován jako číslo se znaménkem, je pro získání správné hodnoty nutné použít tento blok.

Vzhledem k tomu, že procesory se mohou lišit ve způsobu, jakým ukládají vícebajtová čísla (nejčastěji se setkáváme s tzv. little-endian formátem, kde na nižší adrese najdeme méně významný bajt, existují však i procesory s big-endian formátem, kde je to právě naopak), blok UTOI nabízí možnost prohození pořadí bajtů, pokud to již není řešeno na úrovni ovladače. Pro tuto úpravu slouží parametr **SWAP**.

Upozornění: Prohození pořadí bajtů (nastavením **SWAP=on**) typicky řeší problémy s rozdílným uspořádáním bajtů v procesoru pouze pro hodnoty **bits=16** nebo **bits=32**.

Tento blok propaguje kvalitu signálu. Více informací je uvedeno v sekci [1.4](#).

Vstup

u	Vstupní signál bez znaménka	↓-9.22337E+18 ↑9.22337E+18	Large (I64)

Parametr

bits	Počet platných bitů ve vstupním signálu	↓2 ↑64 ⊙16	Long (I32)
SWAP	Otočení pořadí bajtů vstupu		Bool

Výstup

i	Zkonvertovaný (se znaménkem) vstupní signál	Large (I64)

Kapitola 5

ANALOG – Zpracování analogových signálů

Obsah

ABSR0T – Víceotáčkové rozšíření čidla polohy	143
ASW – Přepínač s automatickou volbou vstupu	145
AVG – Filtr: vlečný průměr	147
AVS – Rozběhová jednotka	148
AVSI – Hladká interpolace trajektorie	149
BPF – Filtr: pásmová propust'	152
CMP – Komparátor s hysterezí	153
CNDR – Kompenzátor složité nelinearity	154
DEL – Dopravní zpoždění s inicializací	156
DELM – Dopravní zpoždění	157
DER – Derivace, filtrace a predikce z posledních $n+1$ vzorků . . .	158
EVAR – Vlečná střední hodnota a směrodatná odchylka	159
INTE – Řízený integrátor	160
KDER – Derivace a filtrace vstupního signálu	162
LPF – Filtr: dolní propust'	164
MINMAX – Vlečné minimum a maximum	166
NSCL – Kompenzátor jednoduché nelinearity	167
OSD – Jednokrokové zpoždění	168
RDFT – Vlečná diskrétní Fourierova transformace	169
RLIM – Omezovač strmosti	171
S10F2 – Výběr jednoho ze dvou analogových vstupů	172
SAI – Zabezpečený analogový vstup	175
SEL – Selektor analogového signálu	178
SELQUAD, SELOCT, SELHEXD – Selektory analogového signálu	179

SHIFTOCT – Posuvný registr pro průběžné ukládání hodnot	181
SHLD – Vzorkovač (sample and hold)	183
SINT – Jednoduchý integrátor	185
SPIKE – Filtr pro potlačení poruch ve tvaru úzkých pulzů	186
SSW – Jednoduchý přepínač	188
SWR – Přepínač s rampovou funkcí	189
VDEL – Dopravní zpoždění s proměnnou délkou	190
ZV4IS – Tvarovač vstupního signálu pro potlačení vibrací	191

Knihovna představuje univerzální škálu funkčních bloků určených pro aplikace řízení a zpracování signálů. Obsahuje bloky jako **ASW**, **AVG**, **BPF** a **DEL**, které nabízejí funkce od manipulace se signálem a průměrování po filtrování a složité podmíněné operace, čímž pokrývají široké spektrum požadavků a scénářů.

ABSR0T – Víceotáčkové rozšíření čidla polohy

Symbol bloku

Licence: [ADVANCED](#)



Popis funkce

Blok **ABSR0T** se typicky používá v případech, kdy máme na nějaké hřídeli absolutní čidlo úhlu natočení v rozsahu např. 5° až 355° (popř. -175° až $+175^\circ$), ale potřebujeme řídit pohyb o několik otáček. Blok předpokládá spojitý signál, takže při přechodu z například 355° na 5° předpokládá, že nastala další otáčka a úhel je ve skutečnosti 365° .

Protože v případě dlouhodobého otáčení jedním směrem by došlo k ztrátě přesnosti, je možné vstupem **R1** nastavit výstup **y** zpět do základního intervalu. Pokud je nastaven příznak **RESR** = on, dojde i k vynulování čítače otáček **irev**. V každém případě je však potřeba zároveň resetovat všechny související signály (např. signál **sp** připojeného regulátoru).

Výstup **MPI** (mid-point indicator) detekuje střední polohu čidla, což může být vhodný okamžik k resetování bloku. Výstup **OLI** (off-limits indicator) informuje o tom, že čidlo natočení je v tzv. mrtvém úhlu, kdy neposkytuje platná data.

Tento blok propaguje kvalitu signálu. Více informací je uvedeno v sekci [1.4](#).

Vstup

u	Signál z absolutního snímače polohy	Double (F64)
R1	Reset bloku	Bool

Parametr

lolim	Dolní mez pro údaj z čidla	$\ominus 3.14159265$	Double (F64)
hilim	Horní mez pro údaj z čidla	$\odot 3.14159265$	Double (F64)
tol	Rozsah pro indikaci středové pozice	$\odot 0.5$	Double (F64)
hys	Hystereze pro indikaci středové pozice		Double (F64)
RESR	Příznak pro resetování počítadla otáček		Bool
	off ... resetovat pouze vypočtenou polohu		
	on vynulovat i počítadlo otáček		

Výstup

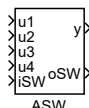
y	Vypočtená poloha	Double (F64)
irev	Počet otáček	Long (I32)

MPI	Indikátor středové polohy	Bool
OLI	Indikátor polohy mimo rozsah senzoru	Bool

ASW – Přepínač s automatickou volbou vstupu

Symbol bloku

Licence: [ADVANCED](#)



Popis funkce

Blok ASW ukládá na výstup y hodnotu jednoho ze vstupů vstup $u1, \dots, u4$ nebo jeden z parametrů $p1, \dots, p4$. Pokud je na vstupu iSW jedna z hodnot $\{1, 2, 3, 4\}$, je na výstupu y hodnota příslušného vstupu. Pokud je na vstupu iSW jedna z hodnot $\{-1, -2, -3, -4\}$, je na výstupu y hodnota příslušného parametru (tj. pro $iSW = -1$ je na výstupu y hodnota $p1$, pro $iSW = 3$ je na výstupu y hodnota $u3$ atd.). Pokud je na vstupu iSW jiná hodnota (tj. $iSW = 0$ nebo $iSW < -4$ nebo $iSW > 4$), je na výstupu y hodnota toho ze vstupů $u1, \dots, u4$ nebo parametrů $p1, \dots, p4$, který se naposledy změnil. Pokud se změní více hodnot najednou, pak se použije hodnota podle následujícího pořadí $p4, p3, p2, p1, u4, u3, u2, u1$. Hodnota se považuje za změněnou, pokud se změnila o více než udává parametr **delta** od minulé detekce změny na příslušném vstupu resp. parametru (tj. změny se uvažují integrálně nikoliv diferenciálně od minulého vzorku). Ve všech režimech je na výstupu oSW číslo vstupu (resp. číslo parametru, pokud je hodnota záporná), který se použil pro generování výstupu y .

Blok ASW má dále tu speciální vlastnost, že nová hodnota y se kopíruje na parametry $p1, \dots, p4$ (stejná vlastnost je i u bloků **PARR**, **PARI**, **PARB**). To má za následek, že všechny externí nástroje jako hodnotu všech těchto vstupů přečtou stejnou hodnotu y . To se hodí zejména v nadřazených systémech, které používají metodu nastav a sleduj (např. "potenciometr" v Iconics Genesis). Tato vlastnost není implementována ve verzi bloku ASW pro Simulink, protože tam není možnost používat externí programy pro čtení vstupu bloku.

POZOR! Pokud je blok zařazen ve schématu v nějaké smyčce, může se stát, že jeden ze vstupů $u1, \dots, u4$ je o krok zpožděn, čímž se zdánlivě ignoruje priorita (výstup oSW pak zcela nepochopitelně signalizuje, že poslední změna nastala na tomto o krok zpožděném vstupu). Dalším důsledkem tohoto stavu je, že externí nástroje na zpožděném vstupu nezobrazují hodnotu y . Takovému chování lze zabránit vhodným použitím bloků **LPBRK** (např. za oba výstupy).

Tento blok propaguje kvalitu signálu. Více informací je uvedeno v sekci [1.4](#).

Vstup

$u1 \dots u4$	Analogový vstupní signál	Double (F64)
iSW	Volba aktivního signálu nebo parametru	Long (I32)

Parametr

delta	Práh pro detekci změny	⊙1e-06	Double (F64)
p1..p4	Parametr pro výběr		Double (F64)

Výstup

y	Zvolený signál nebo parametr		Double (F64)
oSW	Identifikátor použitého vstupu nebo parametru		Long (I32)

AVG – Filtr: vlečný průměr

Symbol bloku

Licence: [STANDARD](#)



Popis funkce

Blok **AVG** počítá vlečný průměr z posledních n vzorků, $n < N$ (N závisí na implementaci), podle vztahu

$$y_k = \frac{1}{n}(u_k + u_{k-1} + \dots + u_{k-n+1}).$$

Není-li ještě k dispozici všech n vzorků (po startu algoritmu), je výstup y nastaven na hodnotu průměru ze všech dostupných vzorků.

Tento blok propaguje kvalitu signálu. Více informací je uvedeno v sekci [1.4](#).

Vstup

u	Vstupní signál filtru	Double (F64)

Parametr

n	Počet vzorků pro průměrování	$\downarrow 1 \uparrow 100000000 \odot 10$	Long (I32)
$n_{\max}$	Rezervovaná paměť pro pole	$\downarrow 10 \uparrow 100000000 \odot 100$	Long (I32)

Výstup

y	Filtrovaný výstupní signál	Double (F64)

AVS – Rozběhová jednotka

Symbol bloku

Licence: **ADVANCED**



Popis funkce

Blok AVS generuje časově optimální trajektorii pohybu z klidové polohy 0 do klidové polohy **sm** při omezení **am** na maximální zrychlení, **dm** na maximální zpomalení a **vm** na maximální rychlost. Při náběžné hraně vstupu **SET** (**off**→**on**) se provede inicializace (výpočet trajektorie) pro aktuální vstupy **am**, **dm**, **vm** a **sm**. Před první inicializací a po dobu inicializace má výstup **RDY** hodnotu **off**, potom **on**. Při náběžné hraně vstupu **START** (**off**→**on**) se spustí generování trajektorie pohybu na výstupech **a**, **v**, **s**, **tt**, přičemž tyto výstupy mají po řadě význam zrychlení, rychlosti, polohy a času. Po dobu generování trajektorie má výstup **BSY** hodnotu **on**, jinak **off**.

Tento blok propaguje kvalitu signálu. Více informací je uvedeno v sekci 1.4.

Vstup

START	Spouštěcí signál (náběžná hrana)	Bool
SET	Inicializace/výpočet trajektorie	Bool
am	Maximální povolené zrychlení [m/s ²]	Double (F64)
dm	Maximální povolené zpomalení [m/s ²]	Double (F64)
vm	Maximální povolená rychlost [m/s]	Double (F64)
sm	Žádaná konečná poloha [m]	Double (F64)

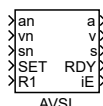
Výstup

a	Zrychlení [m/s ²]	Double (F64)
v	Rychlost [m/s]	Double (F64)
s	Poloha [m]	Double (F64)
tt	Čas [s]	Double (F64)
RDY	Příznak platnosti/připravenosti výstupu	Bool
BSY	Příznak probíhající operace	Bool

AVSI – Hladká interpolace trajektorie

Symbol bloku

Licence: [ADVANCED](#)



Popis funkce

Funkční blok **AVSI** - Akcelerace (A), Rychlost (V), Vzdálenost (S) Interpolace (I) - je navržen pro interpolaci signálů, zejména v aplikacích řízení pohybu. Jeho hlavní účel spočívá v generování hladkých sekvencí polohy (vzdálenosti), rychlosti a zrychlení na základě diskrétních vstupních hodnot. Tento blok je inspirován funkcionalitou z [RM_AxisSpline](#) a nabízí podobné interpolační metody.

Blok **AVSI** přijímá na svých vstupech **sn** (polohu), **vn** (rychlost) a **an** (zrychlení), které jsou generovány externí funkcí (mimo tento blok) s určitou periodou definovanou parametrem **RemTs**. Hodnoty se na vstupu bloku aktualizují na náběžné hraně signálu **SET**.

Interpolace mezi jednotlivými vstupy se provádí s cílem vytvořit plynulé přechody a zajistit kontinuální řízení pohybu nebo jiné aplikace vyžadující regulaci signálu a jeho derivací. Blok podporuje různé interpolační metody určené parametrem **Mode**, které odpovídají možnostem v [RM_AxisSpline](#). Podporované metody jsou:

- **1: linear**: Poloha interpolována lineárně, rychlost jako derivace polohy, zrychlení 0 (tj. rychlost je po částech konstantní funkce se skoky).
- **2: cubic spline**: Poloha je polynom 3. řádu, vypočtený na základě polohy a rychlosti na začátku a konci intervalu; rychlost je derivace polohy, zrychlení derivace rychlosti.
- **3: quintic spline**: Poloha je polynom 5. řádu, vypočtený na základě polohy, rychlosti a zrychlení na začátku a konci intervalu; rychlost je derivace polohy, zrychlení derivace rychlosti.
- **4: cubic approximation (B-spline)**: Poloha je polynom 3. řádu, vypočtený na základě dvou poloh před a dvou poloh za aktuálním intervalem; proložená funkce nemusí přesně procházet zadanými body; rychlost je derivace polohy, zrychlení derivace rychlosti.
- **5: quintic approximation (B-spline)**: Poloha je polynom 5. řádu, vypočtený na základě tří poloh před a tří poloh za aktuálním intervalem; proložená funkce nemusí přesně procházet zadanými body; rychlost je derivace polohy, zrychlení derivace rychlosti.

- **6: all linear:** Poloha, rychlost i zrychlení jsou interpolovány lineárně navzájem nezávisle, tj. rychlost neodpovídá přesně derivaci polohy a zrychlení neodpovídá přesně derivaci rychlosti.
- **7: all cubic:** Poloha i rychlost jsou interpolovány polynomem 3. řádu navzájem nezávisle, tj. rychlost neodpovídá přesně derivaci polohy.
- **8:** *rezervováno pro pozdější použití.*
- **9:** *rezervováno pro pozdější použití.*

Vzhledem k principu činnosti blok **AVSI** zavádí zpoždění signálu, kde aktivní generování hodnot začíná až po dvou kompletních periodách **RemTs** od první náběžné hrany signálu **SET**. Pro B-spline interpolační metody je potřeba větší množství vzorků pro zahájení interpolace.

Blok **AVSI** je primárně určen pro aplikace řízení pohybu, ale lze jej použít i pro jiné typy signálů a jejich derivace. Jeho implementace umožňuje efektivnější a plynulejší přechody mezi jednotlivými stavovými hodnotami bez nutnosti složitého externího řízení.

Při použití je důležité správně nastavit periodu **RemTs** odpovídající generátoru vstupních hodnot a zvolit vhodný **Mode** pro požadovaný typ interpolace.

Tento blok propaguje kvalitu signálu. Více informací je uvedeno v sekci 1.4.

Vstup

an	Zrychlení v příštím (velkém) tiku [unit/s ²]	Double (F64)
vn	Rychlost v příštím (velkém) tiku [unit/s]	Double (F64)
sn	Poloha v příštím (velkém) tiku [unit]	Double (F64)
SET	Vstupy jsou použity při náběžné hraně	Bool
R1	Reset bloku	Bool
dm	Maximální povolené zpomalení při chybě [m/s ²]	Double (F64)

Parametr

RemTs	Perioda pomalejšího (vzdáleného) signálu	⊙0.0	Double (F64)
Mode	Algoritmus pro interpolaci	⊙9	Long (I32)
	1 lineární		
	2 polynom 3. řádu		
	3 polynom 5. řádu		
	4 aproximace polynomem 3.řádu (B-spline)		
	5 aproximace polynomem 5.řádu (B-spline)		
	6 vše lineární		
	7 vše 3.řád		
	8 —		
	9 —		

Výstup

a	Zrychlení [unit/s ²]	Double (F64)
v	Rychlost [unit/s]	Double (F64)
s	Poloha [unit]	Double (F64)
RDY	Příznak platnosti/připravenosti výstupu	Bool
iE	Kód chyby	Bool

BPF – Filtr: pásmová propust

Symbol bloku

Licence: [STANDARD](#)



Popis funkce

Blok BPF realizuje přenos filtru druhého řádu ve tvaru

$$F_s = \frac{2\xi as}{a^2 s^2 + 2\xi as + 1},$$

kde a a ξ jsou po řadě parametry bloku **fm** a **xi**. Parametr **fm** určuje střed frekvenčního pásma propustnosti a **xi** je součinitel relativního tlumení.

Je-li **ISSF** = **on**, potom je stav filtru nastaven do ustáleného stavu okamžitě po spuštění podle první hodnoty vstupu **u**.

Tento blok propaguje kvalitu signálu. Více informací je uvedeno v sekci [1.4](#).

Vstup

u	Vstupní signál filtru	Double (F64)
R1	Reset bloku	Bool
HLD	Pozastavení	Bool

Parametr

fm	Střed pásma propustnosti [Hz]	⊙1.0	Double (F64)
xi	Součinitel relativního tlumení	⊙0.707	Double (F64)
ISSF	Ustálený stav při spuštění		Bool
	off ... nulový počáteční stav		
	on ustálený počáteční stav		

Výstup

y	Filtrovaný výstupní signál	Double (F64)

CMP – Komparátor s hysterezí

Symbol bloku

Licence: [STANDARD](#)



Popis funkce

Blok **CMP** provádí komparaci vstupu **u1** a **u2** s hysterezí **h** podle následujících vztahů:

$$\begin{aligned} Y_{-1} &= 0, \\ Y_k &= \text{hyst}(e_k), \quad k = 0, 1, 2, \dots \end{aligned}$$

kde

$$e_k = u1_k - u2_k$$

a

$$\text{hyst}(e_k) = \begin{cases} 0 & \text{pro } e_k \leq -h \\ Y_{k-1} & \text{pro } e_k \in (-h, h) \\ 1 & \text{pro } e_k \geq h \quad (e_k > h \text{ pro } h = 0) \end{cases}$$

Indexované proměnné odpovídají hodnotám dané veličiny v cyklu, který udává index k , tzn. Y_{k-1} značí hodnotu výstupu v minulém kroku/cyklu. Hodnota Y_{-1} je použita pouze jednou při inicializaci bloku ($k = 0$), pokud je rozdíl vstupních signálů nepřekročí mez hystereze.

Tento blok propaguje kvalitu signálu. Více informací je uvedeno v sekci [1.4](#).

Vstup

u1	První analogový vstup bloku	Double (F64)
u2	Druhý analogový vstup bloku	Double (F64)

Parametr

hys	Hystereze	↓0.0 ⊙0.5 Double (F64)

Výstup

Y	Logický výstupní signál	Bool

CNDR – Kompenzátor složité nelinearity

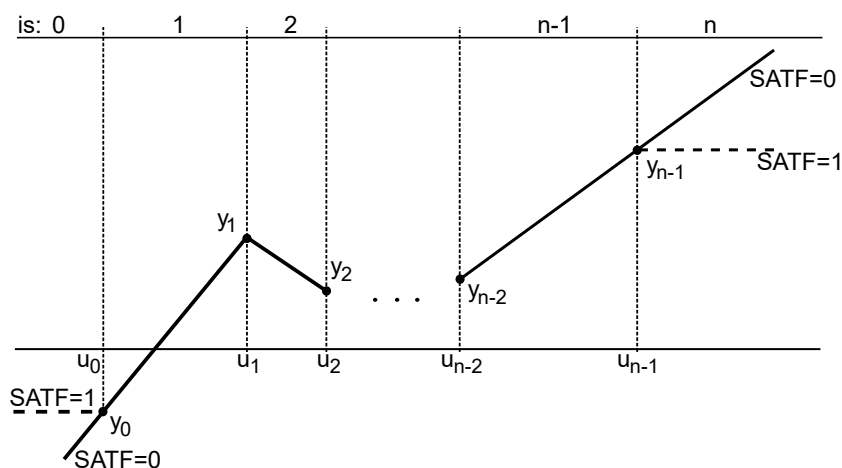
Symbol bloku

Licence: [STANDARD](#)



Popis funkce

Blok CNDR je určen pro kompenzaci složitých nelinearit pomocí po částech lineární transformace zobrazené na níže uvedeném obrázku.



V této souvislosti je důležité upozornit, že v případech $u < u_0$ a $u > u_{n-1}$ je výstup definován v závislosti na parametru SATF.

Tento blok propaguje kvalitu signálu. Více informací je uvedeno v sekci [1.4](#).

Vstup

u Analogový vstupní signál Double (F64)

Parametr

nmax	Rezervovaná paměť pro pole	↓4 ⊙10	Long (I32)
SATF	Saturace	⊙on	Bool
	off ... signál není omezen		
	on saturační meze jsou aktivní		
up	Vektor rostoucích u souřadnic uzlů		Double (F64)
		⊙[0.0 3.9 3.9 9.0 14.5 20.0]	
yp	Vektor y souřadnic uzlů	⊙[0.0 0.0 15.8 38.4 72.0 115.0]	Double (F64)

Výstup

y	Analogový výstupní signál	Double (F64)
is	Sektor nelinearity odpovídající vstupu u	Long (I32)

DEL – Dopravní zpoždění s inicializací

Symbol bloku

Licence: [STANDARD](#)



Popis funkce

Blok DEL realizuje zpoždění vstupního signálu u o n vzorků, tj.

$$y_k = u_{k-n}.$$

Příslušné časové zpoždění je $n \cdot T_S$, kde T_S je perioda spouštění bloku.

Jestliže po spuštění nebo restartu ($R1: \text{off} \rightarrow \text{on} \rightarrow \text{off}$) dosud není zapamatovaných n minulých vzorků ($RDY = \text{off}$), potom

$$y_k = y_0,$$

kde y_0 je inicializační vstup bloku.

Tento blok propaguje kvalitu signálu. Více informací je uvedeno v sekci [1.4](#).

Vstup

u	Analogový vstupní signál	Double (F64)
$R1$	Reset bloku	Bool
y_0	Počáteční hodnota výstupu	Double (F64)

Parametr

n	Zpoždění (počet vzorků)	$\downarrow 0 \uparrow 10000000 \odot 10$	Long (I32)
$n_{\max}$	Rezervovaná paměť pro pole	$\downarrow 10 \uparrow 10000000 \odot 100$	Long (I32)

Výstup

y	Zpožděný vstupní signál	Double (F64)
RDY	Příznak platnosti/připravenosti výstupu	Bool

DELM – Dopravní zpoždění

Symbol bloku

Licence: [STANDARD](#)



Popis funkce

Blok **DELM** realizuje časové zpoždění vstupního signálu **u** o čas, který vznikne zaokrouhlením parametru **del** na nejbližší celočíselný násobek periody T_S spouštění bloku. Po spuštění bloku do času **del** je výstup **y** = 0.

Tento blok propaguje kvalitu signálu. Více informací je uvedeno v sekci [1.4](#).

Vstup

u	Analogový vstupní signál	Double (F64)

Parametr

del	Časové zpoždění [s]	$\odot 1.0$	Double (F64)
nmax	Rezervovaná paměť pro pole	$\downarrow 10 \uparrow 100000000 \odot 100$	Long (I32)

Výstup

y	Zpožděný vstupní signál	Double (F64)

DER – Derivace, filtrace a predikce z posledních $n+1$ vzorků

Symbol bloku

Licence: [STANDARD](#)

Popis funkce

Blok DER prokládá posledních $n + 1$ vzorků ($n \leq N - 1$, N závisí na implementaci) vstupního signálu u přímkou $y = at + b$ metodou nejmenších čtverců. Počátek časové osy je v každém kroku umístěn do aktuálního okamžiku vzorkování vstupu u .

Ze získaných parametrů přímky a a b se počítají v případě $RUN = \text{on}$ výstupy y a z podle vztahů:

$$\begin{aligned} \text{Derivace: } y &= a \\ \text{Filtrace: } z &= b, \text{ pro } t_p = 0 \\ \text{Predikce: } z &= at_p + b, \text{ pro } t_p > 0 \\ \text{Postdikce: } z &= at_p + b, \text{ pro } t_p < 0 \end{aligned}$$

Je-li $RUN = \text{off}$ nebo blok nemá k dispozici posledních $n + 1$ vzorků vstupního signálu ($RDY = \text{off}$), potom $y = 0$, $z = u$.

Tento blok propaguje kvalitu signálu. Více informací je uvedeno v sekci [1.4](#).

Vstup

u	Analogový výstupní signál	Double (F64)
RUN	Povolení běhu algoritmu off ... sledování on filtrace	Bool
t_p	Časový okamžik pro predikci/filtraci	Double (F64)

Parametr

n	Počet vzorků pro interpolaci	$\downarrow 1 \uparrow 100000000 \odot 10$	Long (I32)
$n_{\max}$	Rezervovaná paměť pro pole	$\downarrow 10 \uparrow 100000000 \odot 100$	Long (I32)

Výstup

y	Odhad derivace vstupního signálu	Double (F64)
z	Predikovaný/filtrovaný výstupní signál	Double (F64)
RDY	Příznak platnosti/připravenosti výstupu	Bool

EVAR – Vlečná střední hodnota a směrodatná odchylka

Symbol bloku

Licence: [STANDARD](#)



Popis funkce

Blok **EVAR** počítá střední hodnotu **mu** (μ) a směrodatnou odchylku **si** (σ) z posledních **n** vzorků vstupního signálu **u** podle vztahů

$$\mu_k = \frac{1}{n} \sum_{i=0}^{n-1} u_{k-i}$$

$$\sigma_k = \sqrt{\frac{1}{n} \sum_{i=0}^{n-1} u_{k-i}^2 - \mu_k^2}$$

kde k značí aktuální okamžik vzorkování.

Tento blok propaguje kvalitu signálu. Více informací je uvedeno v sekci [1.4](#).

Vstup

u	Analogový vstupní signál	Double (F64)

Parametr

n	Počet vzorků pro výpočet statistických ukazatelů	Long (I32)
		↓2 ↑100000000 ⊙100
nmax	Rezervovaná paměť pro pole	Long (I32)
		↓10 ↑100000000 ⊙200

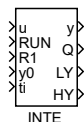
Výstup

mu	Střední hodnota	Double (F64)
si	Směrodatná odchylka	Double (F64)

INTE – Řízený integrátor

Symbol bloku

Licence: [STANDARD](#)



Popis funkce

Blok INTE realizuje řízený integrátor s proměnnou integrační časovou konstantou t_i a indikací dvou úrovní výstupu $y_{\min}$ a $y_{\max}$. Je-li $RUN = \text{on}$ a $R1 = \text{off}$, potom

$$y(t) = \frac{1}{T_i} \int_0^t u(\tau) d\tau + C,$$

kde hodnota $C = y_0$. Je-li $RUN = \text{off}$ a $R1 = \text{off}$, je výstup y zmrazen na jeho poslední hodnotu před sestupnou hranou vstupu RUN . Je-li $R1 = \text{on}$, potom je výstup y resetován na počáteční hodnotu y_0 . Integrace se provádí lichoběžníkovou metodou podle vztahu

$$y_k = y_{k-1} + \frac{T_S}{2T_i}(u_k + u_{k-1}),$$

kde T_S je perioda spouštění bloku. Pokud je $T_i = 0$, místo integrace se provádí sumace podle vztahu

$$y_k = y_{k-1} + u_k.$$

Pro $T_i < 0$ je chování nedefinované.

Pro integraci je také možno použít blok [SINT](#), jehož jednodušší struktura a funkčnost může být pro základní úlohy dostačující.

Tento blok propaguje kvalitu signálu. Více informací je uvedeno v sekci [1.4](#).

Vstup

u	Analogový vstupní signál	Double (F64)
RUN	Povolení běhu algoritmu	Bool
	off ... integrace je pozastavena	
	on ... integrace probíhá	
$R1$	Reset bloku	Bool
y_0	Počáteční hodnota výstupu	Double (F64)
t_i	Integrační časová konstanta	Double (F64)

Parametr

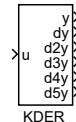
ymin	Definice spodní úrovně	⊙-1.0	Double (F64)
ymax	Definice horní úrovně	⊙1.0	Double (F64)
SAT	Omezení výstupu, když je dosažena krajní úroveň		Bool

Výstup

y	Výstup integrátoru		Double (F64)
Q	Příznak probíhající integrace		Bool
LY	Příznak dosažení spodní meze		Bool
HY	Příznak dosažení horní meze		Bool

KDER – Derivace a filtrace vstupního signálu

Symbol bloku

Licence: **ADVANCED**

Popis funkce

Blok **KDER** je speciálně navržený Kalmanův filtr řádu **norder** tak, aby poskytoval odhady časových derivací řádu 0 až **norder** – 1 lokálně polynomiálních signálů, jejichž měření je zatíženo šumem. Blok je možné využít pro odhad derivací téměř libovolného vstupního signálu $u = u_0(t) + v(t)$ za předpokladu, že užitečný signál $u_0(t)$ a šum $v(t)$ mají odlišné frekvenční spektrum.

Blok se nastavuje pouze pomocí dvou parametrů **pbeta** a **norder**. Parametr **pbeta** je závislý na vzorkovací periodě T_S , frekvenčních vlastnostech vstupního signálu **u** a rovněž frekvenčních vlastnostech a úrovni obsaženého šumu. Platí pro něj přibližný vztah $\text{pbeta} \approx T_S \omega_0$. Pro správnou funkci bloku **KDER** by se frekvenční spektrum vstupního signálu **u** mělo nacházet hluboko pod zlomovou frekvencí filtru ω_0 . Naopak frekvenční spektrum šumů by mělo být co možná nejdále od frekvence ω_0 . Pro vyšší potlačení šumů je nutné volit nižší zlomovou frekvenci ω_0 a tím i parametr **pbeta**.

Druhý parametr **norder** je nutné volit převážně s ohledem na řád odhadovaných derivací. Ve většině případů by mělo stačit použít standardní hodnotu pro 3. řád. Vyšší hodnoty řádu derivačního filtru poskytují o něco lepší odhady derivací nepolynomiálních vstupních signálů za cenu delší doby vysledování (naladění) a vyšších výpočetních nároků.

Tento blok propaguje kvalitu signálu. Více informací je uvedeno v sekci 1.4.

Vstup

u	Vstupní signál filtru	Double (F64)

Parametr

norder	Řád derivačního filtru	↓2 ↑10 ⊙3	Long (I32)
pbeta	Šířka pásma derivačního filtru	↓0.0 ⊙0.1	Double (F64)

Výstup

y	Filtrovaný vstupní signál	Double (F64)
dy	Odhad 1. derivace vstupního signálu	Double (F64)

d2y	Odhad 2. derivace vstupního signálu	Double (F64)
d3y	Odhad 3. derivace vstupního signálu	Double (F64)
d4y	Odhad 4. derivace vstupního signálu	Double (F64)
d5y	Odhad 5. derivace vstupního signálu	Double (F64)

LPF – Filtr: dolní propuť

Symbol bloku

Licence: [STANDARD](#)



Popis funkce

Blok LPF realizuje přenos filtru druhého řádu ve tvaru

$$F_s = \frac{1}{a^2 s^2 + 2\xi a s + 1},$$

kde

$$a = \frac{\sqrt{\sqrt{2}\sqrt{2\xi^4 - 2\xi^2 + 1} - 2\xi^2 + 1}}{2\pi f_b}$$

a **fb** a $\xi = \mathbf{xi}$ jsou parametry bloku. Frekvence **fb** [Hz] určuje šířku pásma filtru a parametr **xi** součinitel relativního tlumení filtru. Útlum na frekvenci **fb** je 3 dB, na $10 \cdot \mathbf{fb}$ přibližně 40 dB. Pro správnou funkci filtru musí platit $f_b < \frac{1}{10T_s}$, kde T_s je perioda spouštění bloku. Doporučená hodnota pro Butterworthův filtr je $\mathbf{xi} = 0,71$ a pro Besselův filtr $\mathbf{xi} = 0,87$.

Je-li **ISSF** = **on**, potom je stav filtru nastaven do ustáleného stavu okamžitě po spuštění podle první hodnoty vstupu **u**.

Tento blok propaguje kvalitu signálu. Více informací je uvedeno v sekci [1.4](#).

Vstup

u	Vstupní signál filtru	Double (F64)
R1	Reset bloku	Bool
HLD	Pozastavení	Bool

Parametr

fb	Šířka pásma [Hz]	⊙1.0	Double (F64)
xi	Součinitel relativního tlumení	⊙0.707	Double (F64)
ISSF	Ustálený stav při spuštění		Bool
	off ... nulový počáteční stav		
	on ustálený počáteční stav		

Výstup

y

Filtrovaný výstupní signál

Double (F64)

MINMAX – Vlečné minimum a maximum

Symbol bloku

Licence: [STANDARD](#)



Popis funkce

Blok MINMAX vyhodnocuje minimum a maximum z posledních **n** vzorků vstupního signálu **u**. Pokud není k dispozici **n** vzorků, je nastaveno **RDY = off** a minimum a maximum se hledá mezi dostupnými vzorky.

Tento blok propaguje kvalitu signálu. Více informací je uvedeno v sekci [1.4](#).

Vstup

u	Analogový vstupní signál	Double (F64)
R1	Reset bloku	Bool

Parametr

n	Počet prvků (délka bufferu)	↓1 ↑10000000 ⊕100	Long (I32)
nmax	Rezervovaná paměť pro pole	↓10 ↑10000000 ⊕200	Long (I32)

Výstup

ymin	Nalezená minimální hodnota	Double (F64)
ymax	Nalezená maximální hodnota	Double (F64)
RDY	Příznak platnosti/připravenosti výstupu	Bool

NSCL – Kompenzátor jednoduché nelinearity

Symbol bloku

Licence: [STANDARD](#)

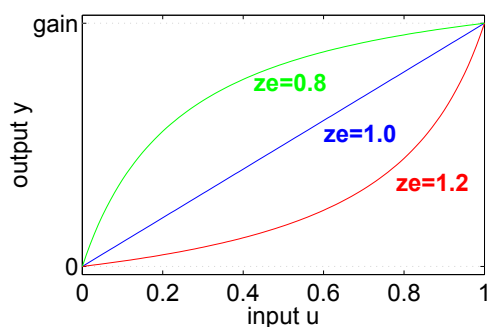


Popis funkce

Blok NSCL kompenzuje v praxi často se vyskytující nelinearity (např. nelinearitu servoventilu) pomocí vztahu

$$y = \text{gain} \frac{u}{ze + (1 - ze)u},$$

kde **gain** a **ze** jsou parametry bloku. Volbou **ze** v intervalu $(0, 1)$ obdržíme konkávní transformaci, zatímco je-li **ze** > 1 , dostaneme transformaci konvexní.



Tento blok propaguje kvalitu signálu. Více informací je uvedeno v sekci [1.4](#).

Vstup

u	Analogový vstupní signál	Double (F64)

Parametr

gain	Zesílení	⊙1.0 Double (F64)
ze	Tvarovací parametr	⊙1.0 Double (F64)

Výstup

y	Analogový výstupní signál	Double (F64)

OSD – Jednokrokové zpoždění

Symbol bloku

Licence: [STANDARD](#)



Popis funkce

Blok **OSD** realizuje jednokrokové zpoždění vstupního signálu **u**. Délka zpoždění (vyjádřena v sekundách) je dána periodou tasku (blíže viz blok [EXEC](#)).

Tento blok propaguje kvalitu signálu. Více informací je uvedeno v sekci [1.4](#).

Vstup

u	Analogový vstupní signál	Any

Parametr

LB	Funkce rozpojení smyčky	Bool

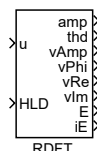
Výstup

y	Analogový výstupní signál	Any

RDFT – Vlečná diskretní Fourierova transformace

Symbol bloku

Licence: [ADVANCED](#)



Popis funkce

Blok **RDFT** počítá diskretní Fourierovu transformaci vstupního signálu pro základní frekvenci **freq** (a případně několik dalších) z posledních **m** vzorků vstupního signálu **u**, kde $m = \text{nper}/\text{freq}/T_S$, tj. z časového okna o délce odpovídající **nper** periodám základní frekvence.

Pokud je **nharm** > 0, je počet vyčíslovaných vyšších harmonických frekvencí dán právě tímto parametrem. Pokud je **nharm** = 0, další vyčíslované frekvence určuje vektorový parametr **freq2**.

Pro každou frekvenci se vyčísluje amplituda (výstup **vAmp**), fáze (výstup **vPhi**), reálná/kosinová složka (výstup **vRe**) a imaginární/sinová složka (výstup **vIm**). Výstupy bloku jsou vektorové, takže obsahují příslušné hodnoty pro všechny analyzované frekvence. Hodnoty pro jednotlivé frekvence se získají pomocí bloků **VTOR**. Výstup **thd** udává celkové harmonické zkreslení, tj. podíl základní a vyšších harmonických frekvencí (jen pokud **nharm** ≥ 1).

Tento blok propaguje kvalitu signálu. Více informací je uvedeno v sekci [1.4](#).

Vstup

u	Analogový vstupní signál	Double (F64)
HLD	Pozastavení	Bool

Parametr

freq	Základní frekvence	↓1e-09 ↑1e+09 ⊙1.0	Double (F64)
nper	Počet period signálu na kterých provádět výpočet	↓1 ↑10000 ⊙10	Long (I32)
nharm	Počet monitorovaných harmonických frekvencí	↓0 ↑16 ⊙3	Long (I32)
ifrunit	Jednotky pro frekvenci	⊙1	Long (I32)
	1 Hz		
	2 rad/s		

iphunit	Jednotky pro fázový posun	⊙1	Long (I32)
	1 stupně		
	2 radiány		
nmax	Rezervovaná paměť pro pole	↓10 ↑100000000 ⊙8192	Long (I32)
freq2	Vektor uživatelem definovaných frekvencí	⊙[2.0 3.0 4.0]	Double (F64)

Výstup

amp	Amplituda základní frekvence	Double (F64)
thd	Celkové harmonické zkreslení	Double (F64)
vAmp	Vektor amplitud pro zadané frekvence	Reference
vPhi	Vektor fázových posunů pro zadané frekvence	Reference
vRe	Vektor reálných částí pro zadané frekvence	Reference
vIm	Vektor imaginárních částí pro zadané frekvence	Reference
E	Příznak chyby	Bool
iE	Kód chyby	Error

RLIM – Omezovač strmosti

Symbol bloku

Licence: [STANDARD](#)



Popis funkce

Blok **RLIM** kopíruje vstup **u** na výstup **y**, avšak maximální dovolená rychlost změny signálu je omezena. Omezení jsou definována časovými konstantami **tp** a **tn** podle následujících vztahů:

$$\begin{aligned} \text{maximální nárůst za sekundu:} & \quad 1/\text{tp} \\ \text{maximální pokles za sekundu:} & \quad -1/\text{tn} \end{aligned}$$

Tento blok propaguje kvalitu signálu. Více informací je uvedeno v sekci [1.4](#).

Vstup

u	Vstupní signál filtru	Double (F64)

Parametr

tp	Časová konstanta určující maximální růst	⊙2.0	Double (F64)
tn	Časová konstanta určující maximální pokles	⊙2.0	Double (F64)

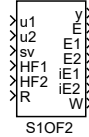
Výstup

y	Filtrovaný výstupní signál	Double (F64)

S10F2 – Výběr jednoho ze dvou analogových vstupů

Symbol bloku

Licence: [ADVANCED](#)



Popis funkce

Blok S10F2 určuje odděleně platnost signálů $u1$ a $u2$ stejným způsobem jako blok SAI. Je-li signál $u1$ (nebo $u2$) neplatný, potom má výstup E1 (nebo E2) hodnotu on a kód chyby je na výstupu iE1 (nebo iE2). Dále se v bloku S10F2 vyhodnocuje odchylka vstupu $u1$ a $u2$ a nastavuje vnitřní příznak D, který má hodnotu on tehdy, jestliže posledních nd vzorků odchylek $|u1 - u2|$ splňuje nerovnost:

$$|u1 - u2| > pdev \frac{vmax - vmin}{100},$$

kde $vmin$ a $vmax$ jsou po řadě dolní a horní mez vstupů $u1$ a $u2$ a $pdev$ je dovolená procentuální odchylka signálů $u1$ a $u2$ z celkového rozsahu. Na základě zjištěné platnosti vstupů (příznaky E1 a E2) a příznaku odchýlení D se určuje zabezpečený výstup y následujícím způsobem:

(i) **Je-li E1 = off a E2 = off a D = off**, pak výstup y je podle parametru $mode$ dán vztahem:

$$y = \begin{cases} \frac{u1+u2}{2}, & \text{pro } mode = 1, \\ \min(u1, u2), & \text{pro } mode = 2, \\ \max(u1, u2), & \text{pro } mode = 3. \end{cases}$$

a výstup ER má hodnotu off, nebyl-li již dříve nastaven na on.

(ii) **Je-li E1 = off a E2 = off a D = on**, potom $y = sv$ a ER = on.

(iii) **Je-li E1 = on a E2 = off (E1 = off a E2 = on)**, potom $y = u2$ ($y = u1$) a výstup ER = off nebyl-li již dříve nastaven na on.

(iv) **Je-li E1 = on a E2 = on**, potom $y = sv$ a ER = on.

Vstup R resetuje vnitřní příznaky chyb F1–F4 (viz. blok SAI) a příznak D. Je-li trvale $R = on$, potom v případě rozpoznání neplatnosti vstupu $u1$ ($u2$) je výstup E1 (E2) nahozen pouze po dobu jednoho cyklu. Naproti tomu při $R = off$ je E1 = on (E2 = on) až do následného resetování (náběžná hrana $R = off \rightarrow on$). Pro výstup ER platí obdobné pravidlo. Je-li trvale $R = on$, pak v případě náběžné hrany vnitřního příznaku D ($off \rightarrow on$) je výstup ER nahozen pouze po dobu jednoho cyklu. Při $R = off$ je nastaveno ER = on až

do následného resetování. Výstup *W* má hodnotu *on* pouze v případech (iii) a (iv), tzn. pokud alespoň jeden z výstupů *E1* a *E2* má hodnotu *on*, tedy pokud je alespoň jeden ze vstupních signálů označen za neplatný.

Parametr *nb* udává počet vzorků po restartu, kdy je potlačeno rozpoznávání platnosti signálů *u1* a *u2*. Parametr *nc* uvádí počet vzorků pro testování neměnnosti (viz blok [SAI](#), podmínka F2). Počet vzorků pro testování variability (viz blok [SAI](#), podmínka F3) je dán parametrem *nr*. Maximální předpokládaná procentuální změna vstupu *u1* (*u2*) z celkového rozsahu *vmax* – *vmin* za *nr* vzorků vstupu *u1* (*u2*) (viz blok [SAI](#)) je určena *prate*. Parametr *nv* vyjadřuje počet vzorků pro testování překročení rozsahu (viz blok [SAI](#), podmínka F4).

Tento blok nepropaguje kvalitu signálu. Více informací je uvedeno v sekci [1.4](#).

Vstup

<i>u1</i>	První analogový vstup bloku	Double (F64)
<i>u2</i>	Druhý analogový vstup bloku	Double (F64)
<i>sv</i>	Náhradní hodnota pro případ chyby	Double (F64)
<i>HF1</i>	Příznak hardwarové chyby vstupu <i>u1</i>	Bool
	off ... vstupní modul signálu pracuje normálně	
	on došlo k hardwarové chybě vstupního modulu	
<i>HF2</i>	Příznak hardwarové chyby vstupu <i>u2</i>	Bool
	off ... vstupní modul signálu pracuje normálně	
	on došlo k hardwarové chybě vstupního modulu	
<i>R</i>	Vynulování vnitřních chybových příznaků	Bool

Parametr

<i>nb</i>	Počet vynechaných vzorků po restartu	⊙10	Long (I32)
<i>nc</i>	Počet vzorků pro testování neměnnosti	⊙10	Long (I32)
<i>nbits</i>	Počet bitů A/D převodníku	⊙12	Long (I32)
<i>nr</i>	Počet vzorků pro testování variability	⊙10	Long (I32)
<i>prate</i>	Maximální procentuální změna	⊙10.0	Double (F64)
<i>nv</i>	Počet vzorků pro testování překročení rozsahu	⊙1	Long (I32)
<i>vmin</i>	Spodní omezení na vstupní signál	⊙-1.0	Double (F64)
<i>vmax</i>	Horní omezení na vstupní signál	⊙1.0	Double (F64)
<i>nd</i>	Počet vzorků pro vyhodnocování odchýlení	⊙5	Long (I32)
<i>pdev</i>	Maximální povolená procentuální odchylka signálů	⊙10.0	Double (F64)
<i>mode</i>	Způsob výpočtu výstupu při platnosti obou vstupů	⊙1	Long (I32)
	1 průměr		
	2 minimum		
	3 maximum		

Výstup

<i>y</i>	Analogový výstupní signál	Double (F64)

E	Indikátor neplatnosti výstupního signálu	Bool
	off ... signál je platný	
	on signál není platný	
E1	Indikátor neplatnosti vstupu u1	Bool
	off ... signál je platný	
	on signál není platný	
E2	Indikátor neplatnosti vstupu u2	Bool
	off ... signál je platný	
	on signál není platný	
iE1	Důvod neplatnosti vstupu u1	Long (I32)
	0 signál je platný	
	1 signál mimo rozsah	
	2 signál se mění příliš málo	
	3 signál se mění jen málo a je mimo rozsah	
	4 signál se mění příliš mnoho	
	5 signál se mění příliš mnoho a je mimo rozsah	
	6 signál se mění příliš málo a příliš mnoho	
	7 signál se mění příliš málo a příliš mnoho a je mimo rozsah	
	8 hardwarová chyba	
iE2	Důvod neplatnosti vstupu u2	Long (I32)
	0 signál je platný	
	1 signál mimo rozsah	
	2 signál se mění příliš málo	
	3 signál se mění jen málo a je mimo rozsah	
	4 signál se mění příliš mnoho	
	5 signál se mění příliš mnoho a je mimo rozsah	
	6 signál se mění příliš málo a příliš mnoho	
	7 signál se mění příliš málo a příliš mnoho a je mimo rozsah	
	8 hardwarová chyba	
W	Varování (neplatný vstupní signál)	Bool
	off ... oba vstupní signály jsou platné	
	on alespoň jeden vstupní signál je neplatný	

SAI – Zabezpečený analogový vstup

Symbol bloku

Licence: [ADVANCED](#)



Popis funkce

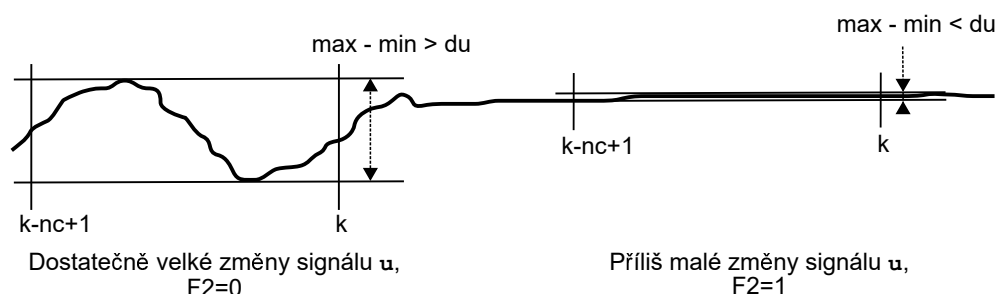
Blok **SAI** testuje vstupní signál u s cílem rozpoznání jeho platnosti. Vstupní signál u se považuje za neplatný (výstup $E = \text{on}$) v následujících případech:

F1: Hardwarová chyba. Vstupní signál $\text{HWF} = \text{on}$.

F2: Vstupní signál u se mění příliš málo. Posledních nc vzorků vstupu u leží v intervalu délky du ,

$$du = \begin{cases} \frac{v_{\max} - v_{\min}}{2^{n_{\text{bits}}}}, & \text{pro } n_{\text{bits}} \in \{8, 9, \dots, 16\} \\ 0, & \text{pro } n_{\text{bits}} \notin \{8, 9, \dots, 16\}. \end{cases}$$

kde $v_{\min}$ a $v_{\max}$ jsou po řadě dolní a horní mez vstupu u a n_{bits} je počet bitů příslušného A/D převodníku. Situace, kdy je splněna podmínka příliš malé změny u , je zobrazena na následujícím obrázku:



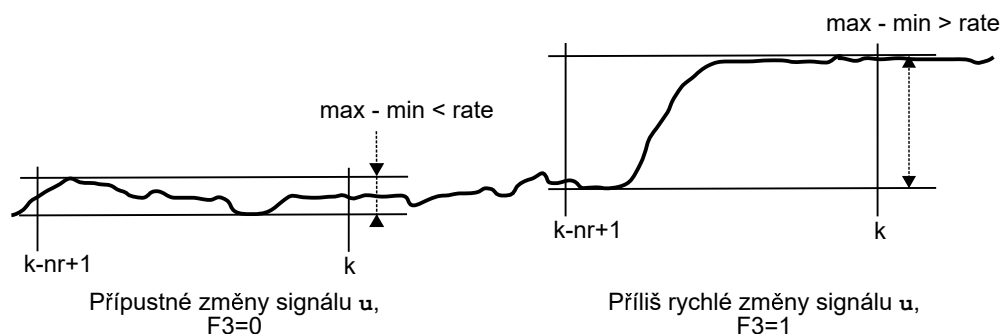
Jestliže je nastaveno $nc = 0$, potom podmínka **F2** není splněna nikdy.

F3: Vstupní signál u se mění příliš rychle. Posledních nr vzorků vstupu u filtrovaného filtrem **SPIKE** neleží v intervalu délky $rate$,

$$rate = prate \frac{v_{\max} - v_{\min}}{100},$$

kde $prate$ vyjadřuje dovolenou procentuální změnu signálu u z celkového rozsahu během nr vzorků (s ohledem na celkový rozsah vstupního signálu $u \in \langle v_{\min}, v_{\max} \rangle$). V bloku je zařazený **SPIKE** filtr s pevnými parametry $mingap = (v_{\max} - v_{\min})/100$ a $q = 2$ odstraňující ze signálu úzké špičky, které by mohly způsobovat nežádoucí splnění této podmínky (blíže viz popis bloku **SPIKE**). Situace, kdy je splněna

podmínka příliš rychlé změny vstupního signálu u , je zobrazena na následujícím obrázku:



Jestliže je nastaveno $nr = 0$, potom podmínka F3 není splněna nikdy.

F4: Vstupní signál u je mimo rozsah. Posledních nv vzorků vstupu u leží mimo přípustný interval $\langle v_{min}, v_{max} \rangle$. Jestliže je nastaveno $nv = 0$, potom podmínka F4 není splněna nikdy.

Je-li signál u platný, potom je beze změny kopírován na výstup y . V opačném případě je do výstupu y dosazena náhradní hodnota ze vstupu sv . V tomto případě má výstup E hodnotu **on** a výstup iE udává kód rozpoznané chyby vstupu u (viz tabulka níže). Vstup R resetuje vnitřní příznaky chyb F1–F4. Je-li trvale $R = \text{on}$, potom v případě rozpoznání neplatnosti vstupu u je výstup E nahozen pouze po dobu jednoho cyklu. Naproti tomu při $R = \text{off}$ je $E = \text{on}$ až do následného resetování (náběžná hrana $R: \text{off} \rightarrow \text{on}$).

Tabulka kódů chyb iE podle vnitřních příznaků F1–F4:

F1	F2	F3	F4	iE
0	0	0	0	0
0	0	0	1	1
0	0	1	0	2
0	0	1	1	3
0	1	0	0	4
0	1	0	1	5
0	1	1	0	6
0	1	1	1	7
1	*	*	*	8

Parametr nb určuje počet vzorků po restartu, kdy je potlačeno rozpoznávání platnosti signálu u . Doporučuje se volit $nb \geq 5$ z důvodu odeznění počátečních podmínek **SPIKE** filtru.

Tento blok nepropaguje kvalitu signálu. Více informací je uvedeno v sekci 1.4.

Vstup

u

Analogový vstupní signál

Double (F64)

sv	Náhradní hodnota pro případ chyby	Double (F64)
HWF	Příznak hardwarové chyby	Bool
	off ... vstupní modul signálu pracuje normálně	
	on došlo k hardwarové chybě vstupního modulu	
R	Vynulování vnitřních chybových příznaků	Bool

Parametr

nb	Počet vynechaných vzorků po restartu	⊙10	Long (I32)
nc	Počet vzorků pro testování neměnnosti	⊙10	Long (I32)
nbits	Počet bitů A/D převodníku	⊙12	Long (I32)
nr	Počet vzorků pro testování variability	⊙10	Long (I32)
prate	Maximální procentuální změna	⊙10.0	Double (F64)
nv	Počet vzorků pro testování překročení rozsahu	⊙1	Long (I32)
vmin	Spodní omezení na vstupní signál	⊙-1.0	Double (F64)
vmax	Horní omezení na vstupní signál	⊙1.0	Double (F64)

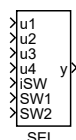
Výstup

y	Analogový výstupní signál	Double (F64)
yf	Výstupní signál filtrovaný SPIKE algoritmem	Double (F64)
E	Indikátor neplatnosti výstupního signálu	Bool
	off ... signál je platný	
	on signál není platný	
iE	Důvod neplatnosti signálu	Long (I32)
	0 signál je platný	
	1 signál mimo rozsah	
	2 signál se mění příliš málo	
	3 signál se mění jen málo a je mimo rozsah	
	4 signál se mění příliš mnoho	
	5 signál se mění příliš mnoho a je mimo rozsah	
	6 signál se mění příliš málo a příliš mnoho	
	7 signál se mění příliš málo a příliš mnoho a je mimo rozsah	
	8 hardwarová chyba	

SEL – Selektor analogového signálu

Symbol bloku

Licence: [STANDARD](#)



Popis funkce

Blok SEL nebude dále podporován, nahraďte jej blokem [SELQUAD](#), [SELOCT](#) nebo [SELHEXD](#). Pozor na změnu významu vstupních signálů *SWn*.

Blok SEL realizuje výběr zvoleného signálu ze čtyř vstupních signálů *u1*, *u2*, *u3* a *u4* a kopíruje ho na výstup *y*. Výběr se provádí podle vstupu *iSW*, jestliže je *BINF* = off nebo podle binárních vstupů *SW1* a *SW2* (*BINF* = on) dle následující tabulky:

iSW	SW1	SW2	y
0	off	off	u1
1	off	on	u2
2	on	off	u3
3	on	on	u4

Tento blok nepropaguje kvalitu signálu. Více informací je uvedeno v sekci [1.4](#).

Vstup

<i>u1</i> .. <i>u4</i>	Analogový vstupní signál	Double (F64)
<i>iSW</i>	Selektor aktivního signálu	Long (I32)
<i>SW1</i>	Binární vstup pro výběr	Bool
<i>SW2</i>	Binární vstup pro výběr	Bool

Parametr

<i>BINF</i>	Výběr pomocí binárních vstupů	Bool
	off ... zakázáno (výběr přes <i>iSW</i>)	
	on povoleno (výběr přes <i>SW1</i> a <i>SW2</i>)	

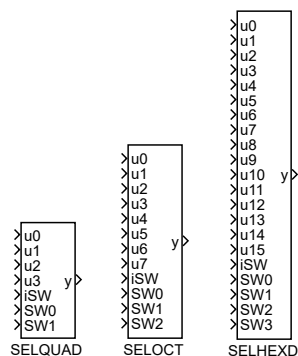
Výstup

<i>y</i>	Zvolený vstupní signál	Double (F64)

SELQUAD, SELOCT, SELHEXD – Selektory analogového signálu

Symboly bloků

Licence: [STANDARD](#)



Popis funkce

Bloky **SELQUAD**, **SELOCT** a **SELHEX** realizují výběr zvoleného signálu ze vstupních signálů a kopírují ho na výstup y. Jediný rozdíl mezi bloky je v počtu vstupních signálů. Výběr aktivního vstupu u0...u15 se provádí podle vstupu iSW, jestliže je BINF = off nebo podle binárních vstupů SW0...SW3 (BINF = on) dle následující tabulky:

iSW	SW0	SW1	SW2	SW3	y
0	off	off	off	off	u0
1	on	off	off	off	u1
2	off	on	off	off	u2
3	on	on	off	off	u3
4	off	off	on	off	u4
5	on	off	on	off	u5
6	off	on	on	off	u6
7	on	on	on	off	u7
8	off	off	off	on	u8
9	on	off	off	on	u9
10	off	on	off	on	u10
11	on	on	off	on	u11
12	off	off	on	on	u12
13	on	off	on	on	u13
14	off	on	on	on	u14
15	on	on	on	on	u15

Tento blok propaguje kvalitu signálu. Více informací je uvedeno v sekci [1.4](#).

Vstup

u0..u15	Analogový vstupní signál	Any
iSW	Selektor aktivního signálu	Long (I32)
SW0..SW3	Binární vstup pro výběr	Bool

Parametr

BINF	Výběr pomocí binárních vstupů	Bool

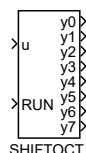
Výstup

y	Zvolený vstupní signál	Any

SHIFTOCT – Posuvný registr pro průběžné ukládání hodnot

Symbol bloku

Licence: [STANDARD](#)



Popis funkce

Blok realizuje funkci posuvného registru s osmi výstupy pro libovolný typ signálů.

Je-li aktivní vstup **RUN**, je v každém tiku algoritmu provedeno následující přiřazení:

$$\begin{aligned} y_i &= y_{i-1}, \quad i = 1..7, \\ y_0 &= u, \end{aligned}$$

tedy hodnota na každém z výstupů **y0** až **y6** je posunuta na výstup v pořadí následující, a hodnota vstupu **u** je přenesena na výstup **y0**.

Blok pracuje s libovolným datovým typem signálu přivedeného na vstup **u**. Požadovaný datový typ je třeba nastavit parametrem **vtype**. Výstupy **y0** až **y7** jsou pak shodného datového typu.

Pokud potřebujete posun dat v registru provádět na základě triggeru, vložte před vstup **RUN** blok [EDGE](#).

Tento blok propaguje kvalitu signálu. Více informací je uvedeno v sekci [1.4](#).

Vstup

u	Analogový vstupní signál	Any
RUN	Povoluje posun výstupů	Bool

Parametr

vtype	Typ výstupů	⊙8 Long (I32)
	1 Bool	
	2 Byte (U8)	
	3 Short (I16)	
	4 Long (I32)	
	5 Word (U16)	
	6 DWord (U32)	
	7 Float (F32)	
	8 Double (F64)	
	10 Large (I64)	

Výstup

y0 . . y7

Analogový výstupní signál

Any

SHLD – Vzorkovač (sample and hold)

Symbol bloku

Licence: [STANDARD](#)



Popis funkce

Blok SHLD je určen pro podržení hodnoty vstupního signálu *u*, přičemž jeho funkce je dána parametrem *mode*.

V případě *vynuceného vzorkování* se nastaví výstup *y* na hodnotu vstupu *u* v okamžiku náběžné hrany (*off*→*on*) řídicího vstupu *SETH* a zůstává konstantní až do příchodu nové náběžné hrany.

Pokud je zvoleno *držení předchozí hodnoty*, na výstup *y* se nastaví poslední hodnota vstupu *u* před příchodem vzestupné hrany na vstupu *SETH*. Tato hodnota je držena po celou dobu, kdy platí *SETH* = *on*. Pokud je na vstupu *SETH* = *off*, je vstup *u* jednoduše kopírován na výstup *y*.

Při režimu *držení aktuální hodnoty* se na výstup *y* nastaví hodnota vstupu *u* v okamžiku náběžné hrany. Tato hodnota je držena po celou dobu, kdy platí *SETH* = *on*. Pokud je na vstupu *SETH* = *off*, je vstup *u* jednoduše kopírován na výstup *y*.

Vstup *R1* slouží k resetování bloku, inicializuje výstup *y* na hodnotu *y0* a má prioritu před vstupem *SETH*.

Zvažte též použití bloku [PARR](#), který může být rovněž použit pro uložení číselné hodnoty.

Tento blok propaguje kvalitu signálu. Více informací je uvedeno v sekci [1.4](#).

Vstup

<i>u</i>	Analogový vstupní signál	Double (F64)
<i>SETH</i>	Vstup pro nastavení a podržení výstupní hodnoty	Bool
<i>R1</i>	Reset bloku	Bool

Parametr

<i>y0</i>	Počáteční hodnota výstupu	Double (F64)
<i>mode</i>	Režim vzorkování	⊙3 Long (I32)
	1 Vynucené vzorkování	
	2 Držení předchozí hodnoty	
	3 Držení aktuální hodnoty	

Výstup

y

Analogový výstupní signál

Double (F64)

SINT – Jednoduchý integrátor

Symbol bloku

Licence: [STANDARD](#)



Popis funkce

Blok **SINT** realizuje diskrétní integrátor popsany diferencní rovnicí

$$y_k = y_{k-1} + \frac{T_S}{2T_i}(u_k + u_{k-1}),$$

kde T_S je perioda spouštění bloku a t_i je integrační konstanta. Pokud je $T_i = 0$, místo integrace se provádí sumace podle vztahu

$$y_k = y_{k-1} + u_k.$$

Pro $T_i < 0$ je chování nedefinované.

Je-li y_k mimo saturační meze **ymin**, **ymax**, potom je výstup i stav integrátoru příslušně omezen.

Pro složitější případy je možné použít blok [INTE](#), který disponuje rozšířenou funkcionalitou.

Tento blok propaguje kvalitu signálu. Více informací je uvedeno v sekci [1.4](#).

Vstup

u	Analogový vstupní signál	Double (F64)

Parametr

ti	Integrační časová konstanta	⊙1.0	Double (F64)
y0	Počáteční hodnota výstupu		Double (F64)
ymax	Horní saturační mez	⊙1.0	Double (F64)
ymin	Dolní saturační mez	⊙-1.0	Double (F64)

Výstup

y	Analogový výstupní signál	Double (F64)

SPIKE – Filtr pro potlačení poruch ve tvaru úzkých pulzů

Symbol bloku

Licence: **ADVANCED**



Popis funkce

Blok **SPIKE** realizuje nelineární filtr odstraňující ze vstupního signálu u izolované úzké špičky (pulzy). Jeden krok **SPIKE** filtru provádí následující transformaci $(u, y) \rightarrow y$:

```
delta := y - u;
if abs(delta) < gap
then
  begin
    y := u;
    gap := gap/q;
    if gap < mingap then gap:= mingap;
  end
else
  begin
    if delta < 0
    then y := y + gap
    else y := y - gap;
    gap := gap * q;
  end
end
```

kde `mingap` a `q` jsou parametry bloku.

Zvolíme-li parametr `mingap` dostatečně velký, potom signál prochází filtrem beze změny. Zmenšováním tohoto parametru je možné docílit stav, kdy dojde k odfiltrování nežádoucích špiček, ale jinak zůstává signál nezkreslen. Doporučená volba je 1 % z celkového rozsahu vstupního signálu u . Parametr `q` určuje rychlost adaptace tolerančního okénka filtru.

Tento blok propaguje kvalitu signálu. Více informací je uvedeno v sekci 1.4.

Vstup

<code>u</code>	Vstupní signál filtru	Double (F64)

Parametr

<code>mingap</code>	Minimální velikost tolerančního okénka	0.01 Double (F64)

q	Rychlost adaptace tolerančního okénka	$\downarrow 1.0 \odot 2.0$	Double (F64)

Výstup

y	Filtrovaný výstupní signál		Double (F64)

SSW – Jednoduchý přepínač

Symbol bloku

Licence: [STANDARD](#)



Popis funkce

Blok SSW provádí výběr jednoho ze dvou vstupních signálů **u1**, **u2** podle logického vstupu **SW** a získanou hodnotu kopíruje na výstup **y**. Je-li **SW = off** (**SW = on**), potom je vybráný vstup **u1** (**u2**).

Tento blok propaguje kvalitu signálu. Více informací je uvedeno v sekci [1.4](#).

Vstup

u1	První analogový vstup bloku	Any
u2	Druhý analogový vstup bloku	Any
SW	Přepínací signál	Bool
	off ... je zvolen vstupní signál u1	
	on je zvolen vstupní signál u2	

Výstup

y	Analogový výstupní signál	Any

SWR – Přepínač s rampovou funkcí

Symbol bloku

Licence: [STANDARD](#)



Popis funkce

Blok **SWR** provádí výběr jednoho ze dvou vstupních signálů **u1**, **u2** podle logického vstupu **SW** a podle něho nastavuje výstup **y**. Je-li **SW = off** (**SW = on**), potom je vybrán vstup **u1** (**u2**). Při přepnutí vstupu se výstup nezmění okamžitě, ale vysleduje vybraný vstup po rampě s definovanou strmostí. Tato strmost může být různá pro oba vstupy **u1**, **u2** a je určena po řadě časovými konstantami **t1** a **t2**. Po vysledování vstupu se funkce omezení strmosti vypne až do následujícího přepnutí.

Tento blok propaguje kvalitu signálu. Více informací je uvedeno v sekci [1.4](#).

Vstup

u1	První analogový vstup bloku	Double (F64)
u2	Druhý analogový vstup bloku	Double (F64)
SW	Přepínací signál	Bool
	off ... je zvolen vstupní signál u1	
	on je zvolen vstupní signál u2	

Parametr

t1	Časová konstanta omezovače strmosti, u2 -> u1	⊙1.0 Double (F64)
t2	Časová konstanta omezovače strmosti, u1 -> u2	⊙1.0 Double (F64)
y0	Počáteční hodnota výstupu	Double (F64)

Výstup

y	Analogový výstupní signál	Double (F64)

VDEL – Dopravní zpoždění s proměnnou délkou

Symbol bloku

Licence: [STANDARD](#)



Popis funkce

Blok VDEL realizuje proměnné časové zpoždění vstupního signálu u o čas přivedený na vstup d . Přesněji, výstupní signál y je zpožděn o čas, který vznikne zaokrouhlením vstupu d na nejbližší celočíselný násobek n periody T_S spouštění bloku. Jestliže po spuštění bloku není dosud k dispozici n posledních vzorků, potom je výstup y nastaven na inicializační hodnotu y_0 .

Tento blok propaguje kvalitu signálu. Více informací je uvedeno v sekci [1.4](#).

Vstup

u	Analogový vstupní signál	Double (F64)
d	Časové zpoždění [s]	Double (F64)

Parametr

y_0	Počáteční hodnota výstupu	Double (F64)
n_{max}	Rezervovaná paměť pro pole	↓10 ↑100000000 ⊖1000 Long (I32)

Výstup

y	Zpožděný vstupní signál	Double (F64)

ZV4IS – Tvarovač vstupního signálu pro potlačení vibrací

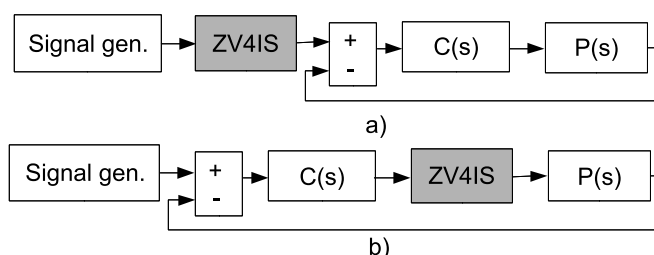
Symbol bloku

Licence: [ADVANCED](#)



Popis funkce

Blok ZV4IS realizuje funkci frekvenčního filtru typu pásmová zádrž. Hlavní oblastí použití je řízení pohybu mechanických systémů s pružnými částmi, kde vlivem nedostatečné tuhosti konstrukce hrozí nebezpečí vzniku reziduálních vibrací. Ty se projevují jako mechanické chvění vybuzené v důsledku momentu nebo síly, kterou aktuátory působí na pracovní mechanismus stroje. Tyto vibrace mohou mít negativní vliv na přesnost regulace, vedou ke zvýšenému opotřebení mechanických částí stroje a v krajním případě mohou způsobit nestabilitu regulačních smyček. Obecně lze tvarovací filtr využít v libovolné aplikaci pro řízení kmitavých systémů nebo pro potlačení konkrétní frekvence ve spojitém signálu.



Tvarovací filtr je možné použít dvěma způsoby. Zapojením v *otevřené smyčce* (viz obrázek výše nahoře) je modifikován referenční signál přicházející od obsluhy nebo od generátoru trajektorie z vyšší úrovně řízení. Výhodou tohoto uspořádání je, že dynamika vlastního filtru neovlivňuje chování podřízené zpětnovazební smyčky. Podmínkou správné funkce je korektní nastavení regulátoru $C(s)$ ve zpětné vazbě, který musí pracovat v lineárním režimu. V opačném případě může dojít ke zkreslení frekvenčního spektra akční veličiny a tím k vybuzení nežádoucích kmitů na rezonančních frekvencích stroje (ve schématu $P(s)$). Záporem přímovazebního zapojení je absence tlumení při působení vnějších poruch. Tuto nevýhodu odstraňuje *zapojení v uzavřené smyčce* (na obrázku dole), kdy filtr je umístěn za zpětnovazební regulátor a tvaruje přímo akční veličinu. V této variantě jsou kompenzovány vibrace vybuzené jak v důsledku změny referenčního signálu tak vlivem působících vnějších poruch. Nevýhodou tohoto zapojení je zanesení dynamického zpoždění do zpětné vazby a tím nutnost přeladit vnitřní regulátor.

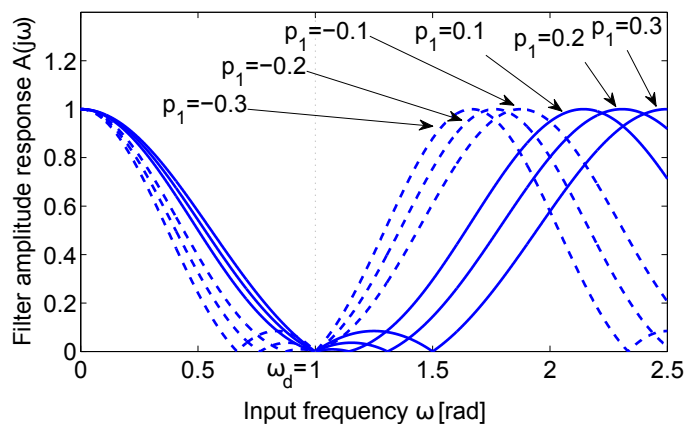
Vlastní algoritmus filtrace lze popsat v časové oblasti vztahem

$$y(t) = A_1 u(t - t_1) + A_2 u(t - t_2) + A_3 u(t - t_3) + A_4 u(t - t_4)$$

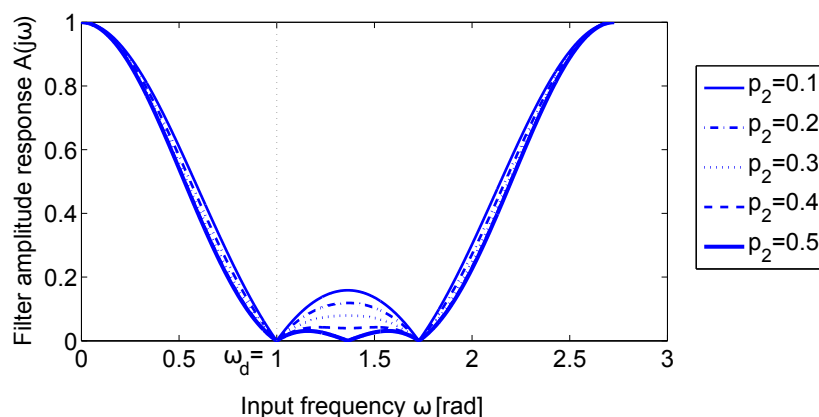
Filtr má tedy strukturu sumy vážených dopravních zpoždění kde zesílení $A_1..A_4$ a hodnoty zpoždění $t_1..t_4$ závisí na volbě typu filtru, frekvenci a tlumení kmitavého módu systému. Výhodou uvedené struktury oproti klasickým dynamickým notch filtrům užívaným v regulační technice je konečná impulzní odezva, která je důležitá zejména v aplikacích řízení pohybu, zaručená stabilita a monotónní přechodová charakteristika filtru a obecně menší zpoždění zaváděné do cesty signálu.

Pro správnou funkci filtru je třeba zadat vlastní frekvenci ω a tlumení ξ kmitavého módu, který má být potlačen. Parametr ipar pak udává typ tvarovacího filtru, pro $\text{ipar} = 1$ je použit jeden z deseti základních filtrů, které se volí parametrem istype . Jednotlivé typy se liší tvarem frekvenční charakteristiky a šířkou nepropustného pásma. Při přesné znalosti ω a ξ je vhodný filtr typu ZV nebo ZVD (Zero Vibration), které dosahují nejrychlejší odezvy na vstupní signál. Při velké neurčitosti v modelu systému/signálu lze použít robustní filtry UEI (Extra Insensitive) nebo UTHEI, které dosahují velké šířky nepropustného pásma za cenu delší odezvy filtru. Číslo na konci názvu filtru odpovídá maximální přípustné procentuální úrovni vybuzeých vibrací pro dané ω a ξ (jedno, dvě nebo pět procent).

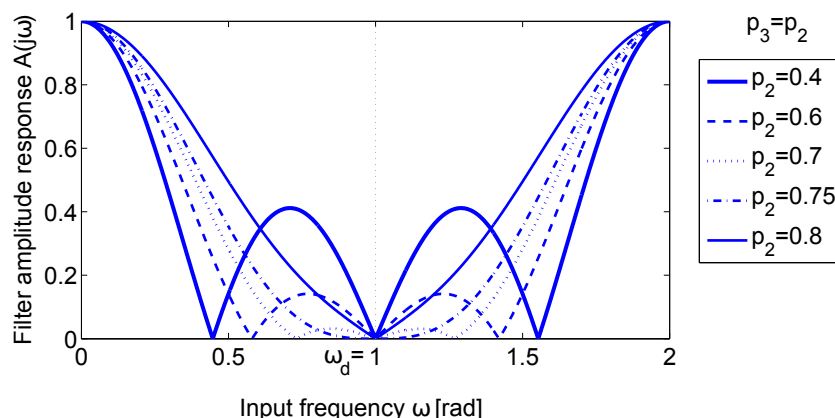
Pro jemné ladění filtru lze použít kompletní parametrizaci volbou $\text{ipar} = 2$, pro kterou se zpřístupní volné parametry p_alpha , p_a2 a p_a3 . Ty určují tvar frekvenční charakteristiky filtru a lze je použít pro nalezení optimálního kompromisu mezi robustností a zavedeným zpožděním.



Parametr asymetrie p_alpha určuje relativní polohu sedla nepropustné oblasti frekvenční charakteristiky filtru vzhledem k nastavené frekvenci ω . Kladná hodnota znamená posun vpravo do oblasti vyšších frekvencí, záporná do opačného směru, nulová hodnota vede na symetrickou charakteristiku (viz obrázek výše). S parametrem p_alpha souvisí také délka filtru, tedy celkové zpoždění zavedené do cesty vstupního signálu, obecně menší hodnota znamená pomalejší filtr s větším zpožděním. Asymetrické filtry jsou vhodné v případech, kdy frekvence, která má být tlumena je proměnná a pohybuje se v určitém intervalu nad nebo pod nominální známou hodnotou, přičemž vyšší pravděpodobnost se předpokládá na jednom z okrajů intervalu (asymetrická hustota pravděpodobnosti).



Parametr necitlivosti p_{a2} určuje šířku a úroveň útlumu nepropustného pásma filtru. Větší hodnota znamená širší nepropustnou oblast s větším tlumením. Pro praktické aplikace je doporučeno nastavit hodnotu $p_{a2} = 0.5$ pro maximální robustnost tvarovacího filtru vůči chybě v modelu systému/signálu.



Doplňkový parametr p_{a3} je nutné nastavit pro symetrické filtry (volba $p_{alpha} = 0$). V tomto případě je pro praktické použití vhodné volit *shodné hodnoty* $p_{a2} = p_{a3}$ na intervalu $< 0, 0.75 >$. Menší hodnoty vedou na rychlejší filtry s úzkým nepropustným pásmem, větší pak na robustní tvarovače s širokým pásmem útlumu a delší odezvou (viz obrázek).

Tento blok propaguje kvalitu signálu. Více informací je uvedeno v sekci 1.4.

Vstup

u Vstupní signál filtru Double (F64)

Parametr

omega Vlastní frekvence 1.0 Double (F64)

xi	Součinitel relativního tlumení		Double (F64)
ipar	Specifikace	⊙1	Long (I32)
	1 základní typy tvarovačů signálu		
	2 kompletní parametrizace		
istype	Typ	⊙2	Long (I32)
	1 ZV		
	2 ZVD		
	3 ZVDD		
	4 MISZV		
	5 UEI1		
	6 UEI2		
	7 UEI5		
	8 UTHEI1		
	9 UTHEI2		
	10 UTHEI5		
p_alpha	Asymetrie tvarovače	⊙0.2	Double (F64)
p_a2	Necitlivost	⊙0.5	Double (F64)
p_a3	Přídavný parametr (pouze pro p_alpha=0)	⊙0.5	Double (F64)
nmax	Rezervovaná paměť pro pole	↓10 ↑10000000 ⊙1000	Long (I32)

Výstup

y	Filtrovaný výstupní signál	Double (F64)
E	Příznak chyby	Bool
	off ... bez chyby	
	on nastala chyba	

Kapitola 6

GEN – Generátory signálů

Obsah

ANLS – Řízený generátor po částech lineární funkce	196
BINS – Řízený generátor binární posloupnosti	198
BIS – Generátor binární posloupnosti	200
BISR – Generátor binární posloupnosti s možností pozastavení	202
MP – Ručně generovaný pulz	204
PRBS – Pseudonáhodná binární posloupnost	205
SG, SGI – Generátory signálu	207

Knihovna GEN je specializována na generování signálů. Obsahuje bloky jako [ANLS](#) pro generování po částech lineární funkce času nebo generátory binárních sekvencí [BINS](#), [BIS](#), [BISR](#). Knihovna také nabízí [MP](#) pro manuální generování pulsních signálů, [PRBS](#) pro generování pseudonáhodných binárních sekvencí a [SG](#) pro generování periodických signálů. Tato knihovna poskytuje zásadní nástroje pro vytváření a manipulaci s různými typy signálů.

ANLS – Řízený generátor po částech lineární funkce

Symbol bloku

Licence: **STANDARD**

Popis funkce

Blok ANLS generuje na výstupu y po částech lineární funkci zadanou uzlovými body $t_1, y_1; t_2, y_2; t_3, y_3; t_4, y_4$. Počáteční hodnota y je definována parametrem y_0 . Start generování funkce (časový okamžik 0) je určen náběžnou hranou (**off**→**on**) vstupu **RUN**. V intervalu $\langle t_i, t_{i+1} \rangle, i = 0, \dots, 3, t_0 = 0$ je výstup y definován vztahem

$$y = y_i + \frac{y_{i+1} - y_i}{t_{i+1} - t_i}(t - t_i).$$

Pro generování skokové změny výstupního signálu je nutné definovat dva uzly ve stejném časovém okamžiku (tj. $t_i = t_{i+1}$). Generování končí, když je dosaženo času t_4 nebo když je dosaženo času t_i a následující uzel předchází aktivnímu uzlu (tj. $t_{i+1} < t_i$). Výstup si poté uchová svou konečnou hodnotu. Pokud je parametr **RPT** nastavený na **on**, pak se místo ponechání konečné hodnoty blok vrátí do výchozího stavu y_0 , vnitřní časovač bloku se nastaví na 0 a sekvence je generována opakovaně. To lze použít ke generování funkcí čtvercových nebo pilových zubů. Generování může být také předčasně ukončeno vstupním signálem **RUN** nastaveným na **off**. V takovém případě se blok vrátí do výchozího stavu y_0 , vnitřní časovač bloku je nastaven na 0 a $is = 0$ se stane aktivním časovým intervalem.

Tento blok nepropaguje kvalitu signálu. Více informací je uvedeno v sekci 1.4.

Vstup

RUN	Povolení běhu algoritmu	Bool

Parametr

y_0	Počáteční hodnota výstupu		Double (F64)
t_1	Čas uzlového bodu 1	⊙1.0	Double (F64)
y_1	Hodnota uzlového bodu 1		Double (F64)
t_2	Čas uzlového bodu 2	⊙1.0	Double (F64)
y_2	Hodnota uzlového bodu 2	⊙1.0	Double (F64)
t_3	Čas uzlového bodu 3	⊙2.0	Double (F64)
y_3	Hodnota uzlového bodu 3	⊙1.0	Double (F64)
t_4	Čas uzlového bodu 4	⊙2.0	Double (F64)

y4	Hodnota uzlového bodu 4	Double (F64)
RPT	Opakování sekvence	Bool
	off ... zakázáno	
	on povoleno	

Výstup

y	Analogový výstupní signál	Double (F64)
is	Index aktivního časového úseku	Long (I32)

BINS – Řízený generátor binární posloupnosti

Symbol bloku

Licence: [STANDARD](#)



Popis funkce

Blok BINS generuje na výstupu Y binární sekvenci podobně jako blok BIS. Binární sekvence je dána parametry bloku.

- Počáteční hodnota výstupu je dána parametrem YO.
- Kdykoli dojde k náběžné hraně (**off**→**on**) na vstupu START (i když se generuje binární sekvence), interní časovač bloku je nastaven na 0 a spuštěn, výstup Y je nastaven na YO.
- Hodnota výstupu se invertuje v časových okamžicích **t1**, **t2**, ..., **t8** (**off**→**on**, **on**→**off**).
- Pro **RPT** = **off** dojde k poslední změně výstupu v čase t_i , kde $t_{i+1} = 0$ a výstup pak udržuje svou hodnotu až do další náběžné hrany (**off**→**on**) na vstupu START.
- Pro **RPT** = **on**, místo poslední změny výstupu, se blok vrátí do počátečního stavu, výstup Y je nastaven na YO, interní časovač bloku je nastaven na 0 a spuštěn. V důsledku toho se binární sekvence generuje opakovaně.

Na rozdíl od bloku BIS se změny parametrů **t1**...**t8** akceptují pouze při náběžné hraně na vstupu START.

Časy přepnutí jsou interně zaokrouhleny na nejbližší celočíselný násobek periody vykonávání, což může vést např. k zániku velmi rychlých pulsů ($< T_S/2$) nebo k sloučení po sobě jdoucích rychlých pulsů do jednoho pomalého pulsu. Proto se doporučuje používat jako časy přepnutí celočíselné násobky periody vykonávání.

Tento blok nepropaguje kvalitu signálu. Více informací je uvedeno v sekci 1.4.

Vstup

START Spouštěcí signál (náběžná hrana)

Bool

Parametr

Y0	Počáteční hodnota výstupu		Bool
	off ... vypnuto/nepravda		
	on zapnuto/pravda		
t1..t8	Okamžik přepnutí [s]	$\downarrow 0.0 \odot 1.0$	Double (F64)
RPT	Opakování sekvence		Bool
	off ... zakázáno		
	on povoleno		

Výstup

Y	Logický výstupní signál	Bool
is	Index aktivního časového úseku	Long (I32)

BIS – Generátor binární posloupnosti

Symbol bloku

Licence: [STANDARD](#)



Popis funkce

Blok BIS generuje na výstupu Y binární sekvenci. Sekvence je dána parametry bloku.

- Počáteční hodnota výstupu je dána parametrem Y0.
- Interní časovač bloku je nastaven na 0 při inicializaci bloku.
- Interní časovač bloku je okamžitě spuštěn při inicializaci bloku.
- Hodnota výstupu se invertuje v časových okamžicích $t_1, t_2, \dots, t_8$ (off→on, on→off).
- Pro $RPT = \text{off}$ dojde k poslednímu přepnutí výstupu v čase t_i , kde $t_{i+1} = 0$ a poté výstup udržuje svou hodnotu na neurčito.
- Pro $RPT = \text{on}$, namísto posledního přepnutí výstupu, se blok vrátí do svého počátečního stavu, výstup Y je nastaven na Y0, interní časovač bloku je nastaven na 0 a spuštěn. V důsledku toho se binární sekvence generuje opakovaně.

Všechny parametry $t_1 \dots t_8$ lze měnit za běhu a všechny změny jsou okamžitě přijímány.

Časy přepnutí jsou interně zaokrouhleny na nejbližší celočíselný násobek periody vykonávání, což může vést např. k zániku velmi rychlých pulsů ($< T_S/2$) nebo k sloučení po sobě jdoucích rychlých pulsů do jednoho pomalejšího pulsu. Proto se doporučuje používat jako časy přepnutí celočíselné násobky periody vykonávání.

Viz také blok [BINS](#), který umožňuje spuštění sekvence externím signálem.

Tento blok nepropaguje kvalitu signálu. Více informací je uvedeno v sekci [1.4](#).

Parametr

Y0	Počáteční hodnota výstupu off ... vypnuto/nepravda on zapnuto/pravda	Bool
$t_1 \dots t_8$	Okamžik přepnutí [s]	↓0.0 ⊙1.0 Double (F64)
RPT	Opakování sekvence off ... zakázáno on povoleno	Bool

Výstup

y	Logický výstupní signál
is	Index aktivního časového úseku

Bool
Long (I32)

BISR – Generátor binární posloupnosti s možností pozastavení

Symbol bloku

Licence: [STANDARD](#)



Popis funkce

Blok BISR generuje na výstupu Y binární sekvenci. Vstup RUN musí být nastaven na **on** po celou dobu trvání sekvence. Když je RUN **off**, sekvence i interní časovač jsou pozastaveny.

Binární sekvence je dána parametry bloku. Počáteční hodnota výstupu je dána parametrem Y0. Hodnota výstupu Y se invertuje (**off**→**on**, **on**→**off**) v časových okamžicích $t_1, t_2, \dots, t_8$. Parametr ADDT definuje, zda jsou časové okamžiky t_i relativní k první náběžné hraně na vstupu RUN nebo relativní k poslednímu přepnutí výstupu Y.

Pokud v požadované binární sekvenci je méně než 8 hran, nastavte libovolný z parametrů t_i na nulu a ostatní budou ignorovány.

Kdykoli dojde k náběžné hraně na vstupu R1, výstup Y je nastaven na Y0 a interní časovač je resetován. Vstup R1 má přednost před vstupem RUN.

Pro $RPT = \text{off}$ dojde k poslednímu přepnutí výstupu v čase t_i , kde $t_i + 1 = 0$ a poté výstup udržuje svou hodnotu až do další náběžné hrany (**off**→**on**) na vstupu START. Pro $RPT = \text{on}$, namísto posledního přepnutí výstupu, se blok vrátí do svého počátečního stavu, výstup Y je nastaven na Y0, interní časovač bloku je nastaven na 0 a spuštěn. V důsledku toho se binární sekvence generuje opakovaně.

Blok BISR je rozšířenou verzí bloku [BINS](#).

Tento blok nepropaguje kvalitu signálu. Více informací je uvedeno v sekci [1.4](#).

Vstup

RUN	Povolení běhu algoritmu	Bool
R1	Reset bloku	Bool

Parametr

Y0	Počáteční hodnota výstupu off ... vypnuto/nepravda on zapnuto/pravda	Bool
ADDT	Postupné časování off ... Absolutní časování (sekvence jako celek) on Postupné časování (segment po segmentu)	Bool
RPT	Opakování sekvence off ... zakázáno on povoleno	Bool

$t_1 \dots t_8$	Okamžik přepnutí [s]	$\downarrow 0.0 \odot 1.0$	Double (F64)

Výstup

Y	Logický výstupní signál	Bool
is	Index aktivního časového úseku	Long (I32)

MP – Ručně generovaný pulz

Symbol bloku

Licence: [STANDARD](#)



Popis funkce

Blok MP generuje na výstupu **Y** pulz délky **pwidth** při náběžné hraně parametru **BSTATE** (**off**→**on**). Hodnota parametru **BSTATE** je algoritmem bloku okamžitě shozena na hodnotu **off** (**BSTATE** představuje signál z krátce stisknutého tlačítka). Je-li **RPTF** = **on**, potom je aktivní opětovné nahození **BSTATE** během generování pulsu a výsledkem je prodloužení generovaného pulsu. Je-li **RPTF** = **off**, je nahození **BSTATE** během generování výstupního pulsu neúčinné, tj. výstupní pulzy mají vždy šířku specifikovanou **pwidth**.

Blok MP reaguje pouze na náběžnou hranu parametru **BSTATE**, nelze ho tedy použít k vygenerování pulsu ihned při startu exekutivy **REXYGEN**. K tomuto účelu použijte blok [BIS](#).

Tento blok nepropaguje kvalitu signálu. Více informací je uvedeno v sekci [1.4](#).

Parametr

BSTATE	Aktivace výstupního pulzu		Bool
	off ... žádná činnost		
	on vygenerování výstupního pulzu		
pwidth	Šířka pulzu [s] (0 = jeden tick)	⊙1.0	Double (F64)
RPTF	Povolení prodloužení pulzu		Bool
	off ... zakázáno		
	on povoleno		

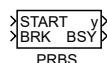
Výstup

Y	Logický výstupní signál	Bool

PRBS – Pseudonáhodná binární posloupnost

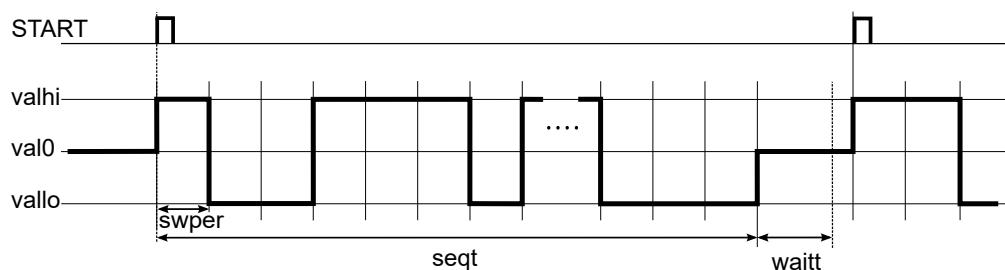
Symbol bloku

Licence: [STANDARD](#)



Popis funkce

Blok PRBS generuje pseudonáhodnou binární posloupnost. Způsob generování je zřejmý z níže uvedeného obrázku.



Počáteční a konečná hodnota posloupnosti je **val0**. Z této hodnoty je generování startováno náběžnou hranou na vstupu **START** (**off**→**on**). V tom okamžiku se výstup **y** přepne z hodnoty **val0** na hodnotu **valhi** a dále se přepíná na druhou možnou hladinu s časovou periodou **swper** a pravděpodobností **swprob**. Tak se postupuje až do uběhnutí času **seqt**. Posloupnost je ukončena opět hodnotou **val0**. Následuje prodleva **waitt** sloužící pro ustálení odezvy řízené soustavy. Teprve poté je možné odstartovat generování nové posloupnosti. V případě potřeby je možné generování posloupnosti přerušit vstupem **BRK = on**.

Tento blok nepropaguje kvalitu signálu. Více informací je uvedeno v sekci [1.4](#).

Vstup

START	Spouštěcí signál (náběžná hrana)	Bool
BRK	Signál pro přerušení	Bool

Parametr

val0	Počáteční a koncová hodnota	Double (F64)
valhi	Horní hladina výstupu y	⊙1.0 Double (F64)
vallo	Dolní hladina výstupu y	⊙-1.0 Double (F64)
swper	Perioda náhodného přepínání výstupu [s]	⊙1.0 Double (F64)
swprob	Pravděpodobnost přepnutí	↓0.0 ↑1.0 ⊙0.2 Double (F64)
seqt	Délka posloupnosti [s]	⊙10.0 Double (F64)

<code>waitt</code>	Doba ustálení [s]	⊙2.0 Double (F64)

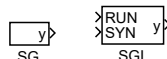
Výstup

<code>y</code>	Vygenerovaná pseudonáhodná binární posloupnost	Double (F64)
<code>BSY</code>	Příznak probíhající operace	Bool

SG, SGI – Generátory signálu

Symboly bloků

Licence: [STANDARD](#)



Popis funkce

Bloky **SG** a **SGI** generují periodické funkce podle nastavení parametru **isig**, který určuje typ signálu: sinusovou vlnu, obdélníkový signál (se střídou 1), pilovitý signál, náhodný signál (bílý šum s rovnoměrným rozdělením) nebo trojúhelníkový signál. Amplitudu a frekvenci výstupního signálu **y** lze nastavit pomocí parametrů **amp** a **freq**. Frekvence může být zadána v jednotkách **Hz** nebo **rad/s**, což se určuje parametrem **ifrunit**. Pro typy signálu sinus, obdélníkový, pilovitý a trojúhelníkový signál (tj. $\text{isig} \in \{1, 2, 3, 5\}$) je možné nastavit fázový posun, který se zadává parametrem **phase** v rozmezí $(0, 2\pi)$. Jednotku fázového posunu (radiány nebo stupně) určuje parametr **iphunit**.

V případě bloku **SGI** je možné navíc synchronizovat počátek generování výstupů u více generátorů **SGI** pomocí vstupů **RUN** a **SYN**. Vstup **SYN** je možné používat za běhu po změně parametrů bloku.

Tento blok nepropaguje kvalitu signálu. Více informací je uvedeno v sekci [1.4](#).

Vstup

RUN	Povolení běhu algoritmu	Bool
SYN	Synchronizační signál	Bool

Parametr

isig	Typ generovaného signálu	⊙1	Long (I32)
	1 SINUS		
	2 ČTVEREC		
	3 PILA		
	4 NÁHODNÝ		
	5 TROJÚHELNÍK		
amp	Amplituda	⊙1.0	Double (F64)
freq	Frekvence	⊙1.0	Double (F64)
phase	Fázový posun		Double (F64)
offset	Hodnota přičtená k výstupu		Double (F64)
ifrunit	Jednotky pro frekvenci	⊙1	Long (I32)
	1 Hz		
	2 rad/s		

iphunit	Jednotky pro fázový posun	⊙1 Long (I32)
	1 stupně	
	2 radiány	
Výstup		
y	Analogový výstupní signál	Double (F64)

Kapitola 7

REG – Bloky pro regulaci

Obsah

ARLY – Relé s předstihem	211
FIWR – Frekvenční identifikace s rekonstruktorem	212
FLCU – Fuzzy regulátor	217
FRID – Identifikace frekvenční charakteristiky	220
I3PM – Identifikace modelu se třemi parametry	224
LC – Derivační kompenzátor	226
LLC – Integračně-derivační kompenzátor	227
LPI – Hodnocení kvality regulační smyčky	228
MCU – Jednotka pro ruční zadávání	230
PIDAT – PID regulátor s reléovým autotunerem	232
PIDE – PID regulátor se statikou	235
PIDGS – PID regulátor s přepínáním sad parametrů	237
PIDMA – PID regulátor s momentovým autotunerem	239
PIDMAX – PID regulátor s rozšířeným momentovým autotunerem	246
PIDU – PID regulátor	253
PIDUI – PID regulátor s parametry na vstupech	257
POUT – Pulzní výstup	259
PRGM – Programátor	260
PSMPC – Prediktivní regulátor typu pulse-step	262
PWM – Pulzně-šířková modulace	266
RLY – Relé s hysterezí	268
SAT – Saturace výstupu s proměnnými mezemi	269
SC2FA – Stavový regulátor systému 2. řádu s autotunerem	271
SCU – Krokový regulátor s polohovou zpětnou vazbou	278
SCUV – Krokový regulátor s rychlostním výstupem	281
SELU – Selektor aktivního regulátoru	285

SMHCC – Regulátor pro procesy s topením a chlazením	287
SMHCCA – Regulátor pro procesy s topením a chlazením s autotunerem	291
SWU – Přepínač vstupu pro vysledování	298
TSE – Třístavový prvek	299

Bloky pro regulaci tvoří nejrozsáhlejší podknihovnu knihovny RexLib a zahrnují bloky od jednoduchých dynamických kompenzátorů, přes bloky pro přepínání regulačních struktur, bloky pro přizpůsobení výstupů akčním členům (krokové regulátory, šířková modulace) až po několik verzí PID (P, I, PI, PD a PID) regulátorů. Mezi regulátory jsou např. blok [PIDGS](#), umožňující za běhu přepínat několik sad parametrů (tzv. *gain scheduling*), [PIDMA](#) s vestavěným *momentovým autotunerem*, blok [PIDAT](#) s vestavěným reléovým autotunerem nebo blok fuzzy regulátoru [FLCU](#), a další.

ARLY – Relé s předstihem

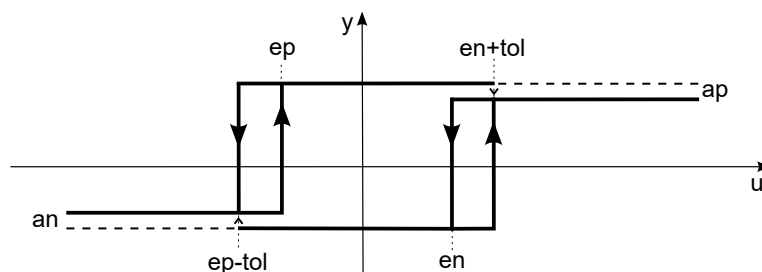
Symbol bloku

Licence: [STANDARD](#)



Popis funkce

Blok **ARLY** je modifikací bloku **RLY**, která umožňuje snížení amplitudy ustálených oscilací v regulačních smyčkách s reléovou zpětnou vazbou. Blok transformuje vstupní signál u na výstupní signál y podle níže uvedeného diagramu.



Tento blok propaguje kvalitu signálu. Více informací je uvedeno v sekci [1.4](#).

Vstup

u	Analogový vstupní signál	Double (F64)

Parametr

ep	Mez pro zapnutí	$\odot -1.0$	Double (F64)
en	Mez pro vypnutí	$\odot 1.0$	Double (F64)
tol	Toleranční mez	$\downarrow 0.0 \odot 0.5$	Double (F64)
ap	Výstup ve stavu ZAPNUTO	$\odot 1.0$	Double (F64)
an	Výstup ve stavu VYPNUTO	$\odot -1.0$	Double (F64)
$y0$	Počáteční hodnota výstupu		Double (F64)

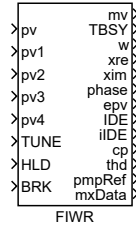
Výstup

y	Analogový výstupní signál	Double (F64)

FIWR – Frekvenční identifikace s rekonstruktorem

Symbol bloku

Licence: [ADVANCED](#)



Popis funkce

Blok **FIWR** realizuje identifikační experiment frekvenční charakteristiky procesu. Soustava je buzena harmonickým signálem se stejnosměrnou složkou **ubias**, amplitudou **uampb** a frekvencí ω postupně probíhající interval $(\mathbf{wb}, \mathbf{wf})$; $\mathbf{wb} < \mathbf{wf}$. Pokud je povolena adaptivní změna amplitudy (**adaptive_amp** = on), pak budící signál nabývá takových hodnot, aby se amplituda výstupu blížila hodnotě **dy_max/2**. Rychlost změny (rozmítání) frekvence je dána stavem **cp** (pokud není adaptivní - viz dále, zadává se do parametru bloku **cpb**), který udává:

- v logaritmickém režimu (**mode** = 1) relativní zmenšení počáteční periody $T_b = \frac{2\pi}{w_b}$ budící sinusovky za čas T_b , tedy:

$$cp = \frac{w_b}{\omega(T_b)} = \frac{w_b}{w_b e^{\gamma T_b}} = e^{-\gamma T_b}.$$

- v lineárním režimu (**mode** = 2) přírůstek frekvence za jednotku času.

Hodnota parametru **cp** se obvykle pohybuje v intervalu $(0, 7; 1)$ v logaritmickém režimu. Pro lineární režim je potřeba parametr volit vzhledem k zadanému frekvenčnímu rozsahu a požadované změně frekvence. Pokud je povolena adaptivní změna rychlosti rozmítání (**adaptive_cp** = on), pak dojde k přepočítání parametru **cp** v závislosti na „chybné“ poloze bodu frekvenční charakteristiky před jeho konvergencí při každém plánovaném zastavení rozmítání (daným vektorem **StopVal**). Aktuální hodnota parametru **cp** je kopírována na výstup **cp**, jeho počáteční hodnotu **cpb** lze zvolit. Minimální a maximální hodnota parametru **cp** **cp_min** a **cp_max** se bere v potaz pouze v režimu **adaptive_cp** = on, jinak se na výstup **cp** kopíruje hodnota **cpb**. V kombinovaném režimu (**mode** = 3) je **cpb** rychlost rozmítání v logaritmických úsecích a lineární úseky včetně rychlostí rozmítání jsou definovány maticí **WL**. Tato matice může být definována:

- **1: Explicitně:** $[w_{b1} \ w_{f1} \ cp_1, w_{b2} \ w_{f2} \ cp_2, \dots, w_{bN} \ w_{fN} \ cp_N]$, kde velikost matice je $[1 \times 3N]$.

- **2: Alternativně:** $[wb_1 \ wb_2 \ \dots \ wb_N; \ wf_1 \ wf_2 \ \dots \ wf_N; \ cp_1 \ cp_2 \ cp_N]$, kde velikost matice je $[3 \times N]$.

wb_i je začátek i -tého lineárního intervalu, wf_i je konec i -tého lineárního intervalu a cp_i je přírůstek frekvence.

Amplituda buzení je v adaptivním módu měněna rovněž plynulým rozmítáním. Poměr parametru rychlosti rozmítání amplitudy ku rychlosti rozmítání frekvence je dán samostatným parametrem **cp_ratio**. Rozmítání amplitudy probíhá vždy logaritmicky a poměr se vztahuje ke skutečné (pro **mode** = 1) nebo teoretické (pro **mode** = 2) logaritmické rychlosti rozmítání frekvence **cp**. Ideální je rozmítat amplitudu rychleji než frekvenci - **cp_ratio** < 1, čímž je zajištěna dostatečně rychlá úprava budící amplitudy pro případné rezonance. Doporučená hodnota je **cp_ratio** $\in \langle 0, 5; 1 \rangle$.

Identifikace systému se spouští nastavením vstupu **TUNE** = **on**. Po uplynutí **rmi** vteřin (předpokládaná doba trvání přechodového děje) startuje výpočet aktuálního bodu frekvenční charakteristiky. Jeho reálná a imaginární část se průběžně kopíruje na výstupy **xre** a **xim**. Fázové zpoždění na dané frekvenci se kopíruje na výstup **phase**. Během procesu identifikace se rozmítání zastaví pokaždé, kdy jsou dosaženy požadované zastavovací body určené v závislosti na parametru **stopBy**: **stopBy** = 1 fázovým zpožděním [*deg*], nebo **stopBy** = 2 frekvencí [*rad/s*]. Doba zastavení souvisí s hodnotami parametrů **q_crit** $\in \langle 0, 001; 100 \rangle$ a **np**. **q_crit** udává prahovou hodnotu relativní změny středních hodnot odhadů dvou po sobě jdoucích okének pro opětovný start rozmítání. Čím menší parametr **q_crit** je, tím přesněji bude bod frekvenční charakteristiky určen, a doba zastavení rozmítání bude delší. Pro kratší vyhodnocovací okénko dané počtem vteřin **np** (obvykle se volí shodně s **rmi**) bude doba zastavení rozmítání kratší. Je možné zvolit maximální počet vyhodnocovacích okének **cmi**, přičemž běžná hodnota je **cmi** = 10. Vstupem **HLD** = **on** je možné ručně zastavit rozmítání frekvence, opětovné nastavení **HLD** = **off** vede k jeho pokračování. V případě potřeby je možné identifikační proces předčasně ukončit vstupem **BRK** = **on**. V takovém případě zůstanou na výstupu **mxData** poslední okamžité hodnoty, na výstupu **pmpRef** všechny přesně změřené body a systém je stále buzen stejnou směrnou složkou **ubias**. Ostatní výstupy jsou nastaveny na hodnotu 0.

Během vlastní identifikace je výstup **TBSY** = 1. Po skončení je nastaven na 0. Při bezchybném experimentu je výstup **IDE** = **off**. Jestliže identifikace skončí chybou, pak je **IDE** = **on** a číslo na výstupu **iIDE** specifikuje příslušnou chybu.

Vektor **StopVal** je možné vložit:

- **1: Explicitně:** $[\phi_1, \phi_2, \dots, \phi_M]$, velikost vektoru je $[1 \times M]$.
- **2: Alternativně:** $[0, 0, \phi_{start}, \phi_{step}, \phi_{end}]$, kde ϕ_{start} je počáteční fáze, ϕ_{step} je krok mezi fázemi a ϕ_{end} je konečná fáze. Velikost vektoru je $[1 \times 5]$.

Pro oba zápisy platí, že vektor fází (pro **stopBy** = 1) musí obsahovat klesající posloupnost a vektor frekvencí (pro **stopBy** = 2) musí obsahovat rostoucí posloupnost.

Parametr **nmax** udává maximální počet prvků vektoru **StopVal**, které je možné vložit. Současně se jedná o maximální šířku matice **pmpRef**, tedy maximální počet přesně změřených bodů, které lze uchovat.

Procesní signál **pv** je rekonstruován pomocí zabudovaného identického rekonstruktoru stavu, který předpokládá na výstupu stejnou frekvenci, jakou má budicí signál. Aby bylo možné odfiltrout drobné nelineární projevy, mohou být rekonstruovány i vyšší harmonické složky s násobností **f2-f4**. V platnost se vždy berou první nenulové násobky. Speciální případ nastává pro dodatečné použití parametru **f5**, pak jsou násobky vždy fixně dané (**f2=2**, **f3=3**, **f4=4**, **f5=5**) bez ohledu na jejich zadané hodnoty. Pro čistě lineární systém mohou v důsledku rekonstrukce neexistujících vyšších harmonických složek některé stavy rekonstruktoru divergovat, pak je zapotřebí nastavit **f2=0**. Rychlost rekonstruktoru lze ladit pozorováním rekonstruované procesní veličiny **epv** prostřednictvím parametru **alpha**, který udává relativní polohu pólů rekonstruktoru.

V rámci jednoho experimentu lze identifikovat naráz až 5 signálů v případě systému s jedním vstupem a více výstupy. Experiment je řízen podle hlavního signálu **pv**, a v adaptivním režimu amplitudy buzení jsou kontrolovány všechny maximálně přípustné hodnoty odezev dané polem **dy_max**. V případě, že délka pole není dostatečná pro **nsig** zvolených signálů ke zpracování, je na všechny nedefinované pozice použita poslední hodnota pole. Jestliže mají všechny signály stejné omezení, stačí tedy zadat pouze jednu hodnotu.

Kód chyby **iIDE**:

- 101 ... vzorkovací frekvence je příliš nízká
- 102 ... saturace výstupu během identifikace
- 103 ... překročena velikost inicializovaných polí
- 104 ... procesní veličina mimo povolený rozsah

- 10 ... varování - vzorkovací frekvence je příliš nízká
- 11 ... varování - **cpb** nevyhovuje nastaveným limitům
- 12 ... varování - rozmítání obnoveno kvůli dosažení max. času
- 13 ... varování - není možné uložit další přesně změřené body
z důvodu vyčerpání inicializované paměti

Tento blok propaguje kvalitu signálu. Více informací je uvedeno v sekci 1.4.

Vstup

pv	Řízená veličina	Double (F64)
TUNE	Zahájení ladicího experimentu	Bool
HLD	Pozastavení	Bool
BRK	Ukončení ladicího experimentu	Bool

Parametr

wb	Počáteční frekvence [rad/s]	↓0.0 ⊙1.0	Double (F64)
wf	Koncová frekvence [rad/s]	↓0.0 ⊙10.0	Double (F64)

cpb	Počáteční rychlost rozmítání	↓0.0 ↑1.0 ⊙0.92	Double (F64)
alpha	Relativní poloha pólů rekonstruktoru	↓0.0 ⊙2.0	Double (F64)
mode	Režim rozmítání	⊙1	Long (I32)
	1 logaritmické		
	2 lineární		
	3 kombinované		
adaptive_cp	Variabilní rychlosti rozmítání	⊙on	Bool
adaptive_amp	Variabilní amplituda buzení	⊙on	Bool
nsig	Počet vstupních signálů	↓1 ↑5 ⊙1	Long (I32)
np	Okno pro detekci konvergence [s]	↓0.0 ⊙3.0	Double (F64)
q_crit	Práh pro vyhodnocení konvergence [%]	↓0.001 ↑100.0 ⊙5.0	Double (F64)
cp_min	Nejmenší rychlost rozmítání	↓0.0 ↑1.0 ⊙0.8	Double (F64)
cp_max	Největší rychlost rozmítání	↓0.0 ↑1.0 ⊙0.99	Double (F64)
cp_ratio	Poměr mezi rychlostí změny frekvence a amplitudy	↓0.0 ↑1.0 ⊙0.8	Double (F64)
uampb	Počáteční amplituda budícího signálu	↓0.0 ⊙1.0	Double (F64)
uamp_max	Nejvyšší amplituda budícího signálu	↓0.0 ⊙5.0	Double (F64)
ubias	Stejnoseměrná složka budícího signálu		Double (F64)
hilim	Největší přípustná hodnota budícího signálu	⊙10.0	Double (F64)
lolim	Nejnižší přípustná hodnota budícího signálu	⊙-10.0	Double (F64)
cmi	Maximální počet oken pro detekci konvergence (pro každou frekvenci)	↓1 ⊙10	Long (I32)
rmi	Čekání na odeznění přechodového děje na startu [s]	↓0.0 ⊙10.0	Double (F64)
f2..f5	Řád (číslo) další odhadované harmonické	↓2 ↑1000 ⊙2	Long (I32)
nmax	Rezervovaná paměť pro pole	↓10 ↑10000000 ⊙40	Long (I32)
stopBy	Ukončení rozmítání při	⊙10	Long (I32)
	1 Fáze [deg]		
	2 Frekvence [rad/s]		
StopVal	Body pro zastavení rozmítání (přesné měření)	⊙[-30 -90 -150]	Double (F64)
WL	Matice lineárních úseků (jen v kombinovaném režimu)	⊙[2.2	Double (F64)
dy_max	Maximální přípustná amplituda měřených sgnálů (pv)	⊙[2.2	Double (F64)

Výstup

mv	Akční zásah regulátoru (manipulated variable)	Double (F64)
TBSY	Příznak probíhajícího ladicího experimentu	Bool
	off ... identifikace neprobíhá	
	on ... běží identifikační experiment	
w	Aktuální frekvence [rad/s]	Double (F64)
xre	Reálná složka frekvenční charakteristiky	Double (F64)
xim	Imaginární složka frekvenční charakteristiky	Double (F64)
phase	Fáze frekvenční charakteristiky [stupně]	Double (F64)
epv	Odhad procesní veličiny	Double (F64)

IDE	Příznak chyby	Bool
iIDE	Kód chyby	Long (I32)
cp	Aktuální rychlost rozmitání	Double (F64)
thd	Celkové harmonické zkreslení	Double (F64)
pmpRef	Přesně změřené body frekvenční charakteristiky (w im re)	Double (F64)
		⊙[2.2
mxData	Průběžně odhadovaný přenos (všechny signály)	⊙[2.2 Double (F64)

FLCU – Fuzzy regulátor

Symbol bloku

Licence: [ADVANCED](#)



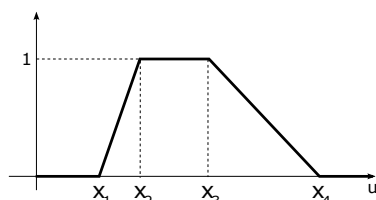
Popis funkce

Blok FLCU realizuje jednoduchý fuzzy regulátor se dvěma vstupy a jedním výstupem. Dostatečný úvod do problematiky fuzzy řízení je uveden v textu [5].

Funkce bloku je jednoznačně určena lichoběžníkovými funkcemi příslušnosti jazykových výrazů vstupů u a v , dále impulsními funkcemi příslušnosti jazykových výrazů k výstupu y a konečně expertními pravidly. Pravidla mají následující tvar:

Jestliže (u je U_i) AND (v je V_j), potom (y je Y_k),

kde $U_i, i = 1, \dots, nu$ jsou jazykové výrazy příslušné ke vstupu u ; $V_j, j = 1, \dots, nv$ jsou jazykové výrazy příslušné ke vstupu v a $Y_k, k = 1, \dots, ny$ jsou jazykové výrazy příslušné k výstupu y . Lichoběžníkové (trojúhelníkové) funkce příslušnosti odpovídající vstupům u a v jsou definovány čtyřmi čísly podle následujícího obrázku



U trojúhelníkových funkcí nejsou všechna čísla $x_1, \dots, x_4$ vesměs různá. Matice funkcí příslušnosti vstupů u a v se potom skládají z řádků $[x_1, x_2, x_3, x_4]$. Matice mf_u a mf_v jsou tedy po řadě typu $(nu \times 4)$ a $(nv \times 4)$.

Impulsní funkce příslušnosti prvního řádu odpovídající výstupu y se zapisují jako trojice

$$y_k, a_k, b_k,$$

kde y_k je hodnota výstupu přiřazená jazykovému výrazu $Y_k, k = 1, \dots, ny$ v případě $a_k = b_k = 0$. Je-li $a_k \neq 0$ a $b_k \neq 0$, potom je výrazu Y_k přiřazena hodnota $y_k + a_k u + b_k v$. Matice funkcí příslušnosti výstupu sty je typu $(ny \times 3)$ a skládá se po řadě z řádků $[y_k, a_k, b_k], k = 1, \dots, ny$.

Soubor pravidel se skládá též jako matice a její řádky jsou $[i_l, j_l, k_l, w_l], l = 1, \dots, nr$, kde i_l, j_l a k_l označuje jistý jazykový výraz příslušný po řadě vstupu u, v a výstupu y . Číslo w_l udává váhu pravidla v procentech $w_l \in \{0, 1, \dots, 100\}$. Tímto způsobem lze jednoduše některé pravidlo zdůraznit, popřípadě vypustit.

Tento blok propaguje kvalitu signálu. Více informací je uvedeno v sekci [1.4](#).

Vstup

u	První analogový vstup bloku	Double (F64)
v	Druhý analogový vstup bloku	Double (F64)

Parametr

umax	Horní omezení vstupu u	$\odot 1.0$	Double (F64)
umin	Dolní omezení vstupu u	$\odot -1.0$	Double (F64)
vmax	Horní omezení vstupu v	$\odot 1.0$	Double (F64)
vmin	Dolní omezení vstupu v	$\odot -1.0$	Double (F64)
nmax	Rezervovaná paměť pro pole	$\downarrow 4 \uparrow 10000 \odot 10$	Long (I32)
mfu	Matice funkcí příslušnosti - vstup u		Double (F64)
		$\odot [-1 \ -1 \ -1 \ 0; \ -1 \ 0 \ 0 \ 1; \ 0 \ 1 \ 1 \ 1]$	
mfv	Matice funkcí příslušnosti - vstup v		Double (F64)
		$\odot [-1 \ -1 \ -1 \ 0; \ -1 \ 0 \ 0 \ 1; \ 0 \ 1 \ 1 \ 1]$	
sty	Matice funkcí příslušnosti - výstup y		Double (F64)
		$\odot [-1 \ 0 \ 0; \ 0 \ 0 \ 0; \ 1 \ 0 \ 0]$	
rls	Matice pravidel	$\odot [1 \ 2 \ 3 \ 100; \ 1 \ 1 \ 1 \ 100; \ 1 \ 0 \ 3 \ 100]$	Byte (U8)

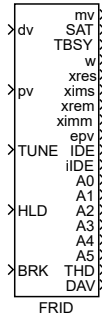
Výstup

y	Analogový výstupní signál	Double (F64)
ir	Dominantní pravidlo	Long (I32)
wr	Stupeň pravdivosti dominantního pravidla	Double (F64)

FRID – Identifikace frekvenční charakteristiky

Symbol bloku

Licence: **ADVANCED**



Popis funkce

Blok **FRID** je určený pro experimentální identifikaci frekvenční odezvy lineárních a slabě nelineárních systémů. Základní princip funkce spočívá ve využití generátoru budícího signálu v podobě rozmítané harmonické funkce s časově proměnnou frekvencí, amplitudou a stejnosměrnou složkou. Tento signál je injektován na vstup identifikovaného systému v otevřené nebo uzavřené smyčce. Na základě pozorované výstupní odezvy systému je pomocí interně implementovaného rekonstruktoru stavu vypočten neparametrický model v podobě bodů frekvenčního přenosu systému.

Jeho primární aplikace je zaměřena v oblasti systémů *motion control* s kmitavou dynamikou, kde je přesná identifikace frekvenčních charakteristik klíčová pro optimální návrh algoritmů řízení.

Identifikační procedura je spuštěna při detekci náběžné hrany **off**→**on** na vstupu **TUNE**. Procedura přepíná mezi dvěma režimy: rozmítáním a měřením. V prvním režimu blok **FRID** generuje na výstupu bloku **mv** rozmítaný sinusový signál jako buzení pro testovaný systém ve tvaru:

$$mv(t) = A(t) \sin\left\{\int_0^t \omega(\tau) d\tau\right\} + u_0(t),$$

kde $A(t)$ je aktuální amplituda, ω aktuální úhlová frekvence a $u_0(t)$ stejnosměrná složka. Pokud je nastaveno **ADAPT_EN** na **on**, amplituda a stejnosměrná složka jsou dynamicky upravovány pro dosažení optimální odezvy systému napříč celým spektrem pozorovaných frekvencí. Toto je důležité zejména u systémů s kmitavou dynamikou, kde se zesílení v oblasti rezonancí a antirezonancí může podstatně měnit až o několik řádů. Počáteční hodnoty jsou dány parametrem **uamp** pro amplitudu a **ubias** pro stejnosměrnou složku. Rychlost změny amplitudy je dána pomocí **adapt_rc**.

Režim rozmítání frekvence je dán parametrem **isweep**. Rychlost změny (rozmítání) frekvence je dána parametrem **cp**, který udává relativní zmenšení počáteční periody

$T_b = \frac{2\pi}{\omega b}$ budící sinusovky za čas T_b , tedy

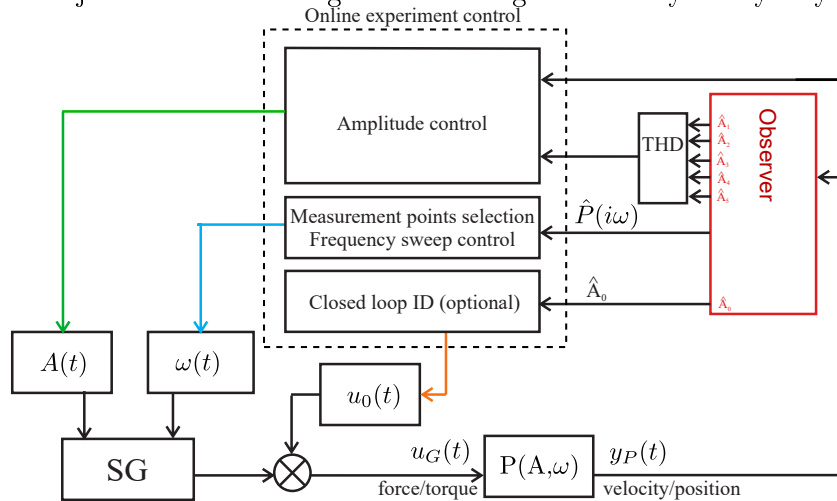
$$c_p = \frac{\omega b}{\omega(T_b)} = \frac{\omega b}{\omega b e^{\gamma T_b}} = e^{-\gamma T_b}.$$

Hodnota parametru c_p se obvykle pohybuje v intervalu $c_p \in (0,95; 1)$.

Pro dané hodnoty ω je rozmítání pozastaveno a identifikace přepne do fáze měření a blok naměří bod frekvenční charakteristiky. V závislosti na nastavení parametru **immode** mohou být hodnoty ω určeny manuálně, rozsahem frekvencí nebo automaticky (viz popis parametru níže). Tento proces usnadňuje identifikaci klíčových dynamických vlastností, včetně (anti)rezonančních frekvencí.

Jádro algoritmu bloku **FRID** je založeno na rekurzivní diskretizaci Kalmanova filtru, což optimalizuje výpočetní efektivitu při zachování vysoké přesnosti odhadu frekvenčního přenosu. Blok dynamicky upravuje parametry excitačního signálu během experimentu identifikace, aby zmírnil efekty nelinearity a dosáhl spolehlivé reprezentace dynamiky systému lineárním modelem frekvenčního přenosu. Adaptace amplitudy zároveň slouží k udržení výstupu v uživatelem definovaných mezích. Rychlost výsledování pozorované výstupní odezvy lze ovlivnit parametrem **obw**, který ovlivňuje šířku pásma rekonstruktoru. Větší hodnota vede na rychlejší výsledování a ustálení odhadu frekvenčního přenosu za cenu většího zesílení šumu měření.

Níže je schématické znázornění bloku **FRID** v rámci sestavy pro identifikaci systému. Diagram ilustruje interakci bloku s generátorem signálu a analyzovaným systémem.



Blok **FRID** nabízí výstupy, které obsahují podrobné informace o frekvenční charakteristice testovaného systému. Výstupy **A1** až **A5** poskytují odhady amplitud prvních pěti harmonických složek Fourierova rozvoje odezvy systému pro danou frekvenci v ustáleném stavu. Tyto amplitudy jsou klíčové pro pochopení, jak systém reaguje na různé frekvence vstupního signálu, a umožňují identifikovat charakteristické vlastnosti systému, jako jsou rezonanční frekvence. Zároveň slouží k detekci nelinearity.

Výstup **THD**, neboli celkové harmonické zkreslení, je ukazatelem míry, do jaké míry se odezva systému odchyľuje od ideální lineární odezvy. Nízká hodnota **THD** naznačuje, že systém se chová převážně lineárně vůči vstupnímu signálu, zatímco vysoká hodnota **THD**

může indikovat přítomnost nelinearity, jako je saturace nebo mechanické vůle. Kromě nelinearity je tímto zohledněn i vliv šumu měření. Hodnota THD je vypočítána jako poměr RMS (root mean square) amplitud vyšších harmonických složek k RMS amplitudě první harmonické složky, což poskytuje ucelený pohled na kvalitu a linearitu odezvy systému.

Více detailů o fungování bloku je možné nalézt v publikaci [6].

Tento blok propaguje kvalitu signálu. Více informací je uvedeno v sekci 1.4.

Vstup

dv	Proměnná dopředné vazby	Double (F64)
pv	Řízená veličina	Double (F64)
TUNE	Zahájení ladicího experimentu	Bool
HLD	Pozastavení	Bool
BRK	Ukončení ladicího experimentu	Bool

Parametr

ubias	Stejnoseměrná složka budicího signálu	Double (F64)
uamp	Amplituda budicího signálu	⊙1.0 Double (F64)
wb	Počáteční frekvence [rad/s]	⊙1.0 Double (F64)
wf	Koncová frekvence [rad/s]	⊙10.0 Double (F64)
isweep	Režim rozmítání	⊙1 Long (I32)
	1 logaritmické	
	2 lineární	
	3 kombinované	
cp	Rychlost rozmítání	⊙0.995 Double (F64)
iavg	Počet hodnot pro průměrování	⊙10 Long (I32)
obw	Šířka pásma	⊙2 Long (I32)
	1 Malá	
	2 Střední	
	3 Velká	
stime	Doba ustálení [s]	⊙10.0 Double (F64)
umax	Maximální amplituda generátoru	⊙1.0 Double (F64)
thdmin	Minimální požadované zkreslení	⊙0.1 Double (F64)
adapt_rc	Rychlost změny amplitudy rozmítání	⊙0.001 Double (F64)
pv_max	Maximální požadovaná amplituda PV	⊙1.0 Double (F64)
pv_sat	Maximální dovolená amplituda PV	⊙2.0 Double (F64)
ADAPT_EN	Zapnutí adaptace amplitudy generátoru	⊙on Bool
immode	Režim měření	⊙1 Long (I32)
	1 Ruční volba frekvencí	
	2 Lineárně nmw frekvencí v intervalu <wb,wf>	
	3 Logaritmicky nmw frekvencí v intervalu <wb,wf>	
	4 Automatická detekce důležitých frekvencí (N/A)	
nwm	Počet frekvencí v automatickém režimu	Long (I32)

wm	Pole frekvencí pro ruční režim [array of rad/s] ⊙[2.0 4.0 6.0 8.0]	Double (F64)

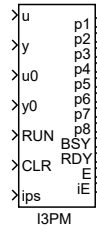
Výstup

mv	Akční zásah regulátoru (manipulated variable)	Double (F64)
SAT	Saturace	Bool
TBSY	Příznak probíhajícího ladicího experimentu off ... identifikace neprobíhá on běží identifikační experiment	Bool
w	Frekvenční charakteristika - frekvence [rad/s]	Double (F64)
xres	Frekvenční charakteristika - reálná složka (rozmítání)	Double (F64)
xims	Frekvenční charakteristika - imaginární složka (rozmítání)	Double (F64)
xrem	Frekvenční charakteristika - reálná složka (změřená)	Double (F64)
ximm	Frekvenční charakteristika - imaginární složka (změřená)	Double (F64)
epv	Odhad PV	Double (F64)
IDE	Příznak chyby	Bool
iIDE	Kód chyby	Long (I32)
DC	Odhad stejnosměrné složky přenosu	Double (F64)
A1..A5	Odhad amplitudy n-té harmonické	Double (F64)
THD	Celkové harmonické zkreslení	Double (F64)
DAV	Platná data	Bool

I3PM – Identifikace modelu se třemi parametry

Symbol bloku

Licence: [ADVANCED](#)



Popis funkce

Blok I3PM identifikuje tříparametrový model soustavy metodou zobecněných momentů. Na rozdíl od bloku [PIDMA](#) je nutné řídit identifikační experiment samostatně. Blok poskytne odhadnuté parametry na výstupy **p1-6** v jednom z následujících tvarů v závislosti na hodnotě vstupu **ips**:

- **0**: model prvního řádu s dopravním zpožděním daným přenosem ve tvaru $F_{FOPDT}(s) = \frac{K}{\tau s + 1} \cdot e^{-Ds}$, kde **p1** = K , **p2** = D , **p3** = τ ,
- **1**: momenty vstupů a výstupů, kde **p1** = mu_0 , **p2** = mu_1 , **p3** = mu_2 , **p4** = my_0 , **p5** = my_1 , **p6** = my_2 ,
- **2**: momenty procesu, kde **p1** = mp_0 , **p2** = mp_1 , **p3** = mp_2 ,
- **3**: charakteristická čísla procesů, kde **p1** = κ , **p2** = μ , **p3** = σ^2 , **p4** = σ . Pro více informací o charakteristických číslech viz dokumentaci bloku [PSMPC](#).

Výstupy **p7** a **p8** jsou rezervovány pro pozdější použití.

Tento blok nepropaguje kvalitu signálu. Více informací je uvedeno v sekci [1.4](#).

Vstup

u	Vstup identifikované soustavy	Double (F64)
y	Výstup identifikované soustavy	Double (F64)
u0	Ustálená hodnota vstupu	Double (F64)
y0	Ustálená hodnota výstupu	Double (F64)
RUN	Spuštění identifikace	Bool
CLR	Nulování bloku	Bool
ips	Význam výstupních signálů	Long (I32)
	0 model prvního řádu s dopravním zpožděním	
	1 momenty vstupu a výstupu	
	2 momenty procesu	
	3 charakteristická čísla procesu	

Parametr

<code>tident</code>	Délka identifikace [s]	⊙100.0 Double (F64)
<code>irtype</code>	Typ regulátoru	⊙6 Long (I32)
	1 D	
	2 I	
	3 ID	
	4 P	
	5 PD	
	6 PI	
	7 PID	
<code>ispeed</code>	Požadovaná rychlost uzavřené smyčky	⊙2 Long (I32)
	1 pomalá uzavřená smyčka	
	2 středně rychlá uzavřená smyčka	
	3 rychlá uzavřená smyčka	

Výstup

<code>p1..p8</code>	Identifikovaný parametr	Double (F64)
<code>BSY</code>	Příznak probíhající operace	Bool
<code>RDY</code>	Příznak platnosti/připravenosti výstupu	Bool
<code>E</code>	Příznak chyby	Bool
	off ... bez chyby	
	on nastala chyba	
<code>iE</code>	Kód chyby	Long (I32)
	1 předčasné ukončení	
	2 $\mu_0=0$	
	3 $m_p=0$	
	4 $\sigma^2 < 0$	

LC – Derivační kompenzátor

Symbol bloku

Licence: [STANDARD](#)



Popis funkce

Blok LC realizuje diskretní simulátor přenosu derivačního članku

$$C(s) = \frac{td \cdot s}{\frac{td}{nd} \cdot s + 1},$$

kde **td** je derivační konstanta a **nd** je parametr určující vliv parazitního filtru prvního řádu. Doporučená hodnota **nd** je $2 \leq nd \leq 10$. Je-li **ISSF** = **on**, potom je stav parazitního filtru nastaven do ustáleného stavu okamžitě po spuštění podle první hodnoty vstupu **u**.

Pro diskretizaci přenosu $C(s)$ je použita přesná diskretizace v okamžicích vzorkování.

Tento blok propaguje kvalitu signálu. Více informací je uvedeno v sekci [1.4](#).

Vstup

u	Analogový vstupní signál	Double (F64)
R1	Reset bloku	Bool
HLD	Pozastavení	Bool

Parametr

td	Derivační časová konstanta	⊙1.0	Double (F64)
nd	Parametr filtru derivační složky	⊙10.0	Double (F64)
ISSF	Ustálený stav při spuštění		Bool
	off ... nulový počáteční stav		
	on ustálený počáteční stav		

Výstup

y	Analogový výstupní signál	Double (F64)

LLC – Integračně-derivační kompenzátor

Symbol bloku

Licence: [STANDARD](#)



Popis funkce

Blok LLC realizuje diskretní simulátor přenosu integračně-derivačního članku

$$C(s) = \frac{a \cdot \tau \cdot s + 1}{\tau \cdot s + 1},$$

kde τ je časová konstanta jmenovatele a její a -násobek ($a \cdot \tau$) je časová konstanta čitatele. Je-li **ISSF** = **on**, potom je stav integračního članku nastaven do ustáleného stavu okamžitě po spuštění podle první hodnoty vstupu u .

Pro diskretizaci přenosové funkce $C(s)$ se používá přesná diskretizace v okamžicích vzorkování. Vzorkovací perioda použitá pro diskretizaci je ekvivalentní k prováděcí periodě bloku LLC.

Tento blok propaguje kvalitu signálu. Více informací je uvedeno v sekci [1.4](#).

Vstup

u	Analogový vstupní signál	Double (F64)
R1	Reset bloku	Bool
HLD	Pozastavení	Bool

Parametr

tau	Časová konstanta	⊙1.0 Double (F64)
a	Koeficient pro výpočet časové konstanty čitatele	Double (F64)
ISSF	Ustálený stav při spuštění	Bool
	off ... nulový počáteční stav	
	on ustálený počáteční stav	

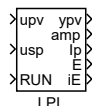
Výstup

y	Analogový výstupní signál	Double (F64)

LPI – Hodnocení kvality regulační smyčky

Symbol bloku

Licence: **ADVANCED**



Popis funkce

Funkční blok LPI (*Loop Performance Index*) je navržen pro ohodnocení kvality zpětné regulace tím, že ovlivňuje hodnotu signálu před jeho přivedením do regulátoru a měří odezvu systému. Tento blok je užitečný pro analýzu a identifikaci chování regulační smyčky v reálném čase.

Procesní veličina **pv** regulační smyčky je připojena na vstup **upv** a z výstupu **ypv** do regulátoru. Požadovaná veličina **sp** regulační smyčky je přivedena na vstup **usp**. Blok se aktivuje signálem **RUN** pouze v automatickém režimu regulátoru, když je žádoucí provést identifikaci regulační smyčky.

Při aktivaci (**RUN**=1) blok LPI injektuje do procesní veličiny sinusový signál o definované amplitudě **ad** a frekvenci **fd**, což umožňuje měření odezvy systému. Výstupní signál je dále zpracován BandPass filtrem a Fourierovou transformací pro určení průměrné amplitudy signálu. Výsledný index výkonnosti **Ip** je vypočítán na základě poměru mezi nastavenými parametry a naměřenou amplitudou, což poskytuje kvantitativní hodnocení potlačení výstupních poruch regulačním systémem.

Výstup **Ip** reflektuje, jak efektivně regulační systém potlačuje poruchy v definovaném frekvenčním pásmu **fa**. Hodnota **Ip**=1 naznačuje, že systém potlačuje poruchy v souladu s očekáváním; hodnoty vyšší než 1 ukazují na lepší výkon; nižší hodnoty indikují horší nastavení regulační smyčky.

Tento blok propaguje kvalitu signálu. Více informací je uvedeno v sekci 1.4.

Vstup

upv	Vstup řízené veličiny	Double (F64)
usp	Vstup požadované hodnoty	Double (F64)
RUN	Povolení běhu algoritmu	Bool

Parametr

ms	Horní omezení citlivostní funkce	↓1.00001 ↑1000.0 ⊙2.0	Double (F64)
fa	Přípustná šířka pásma	↓1e-10 ↑1e+10 ⊙10.0	Double (F64)
fd	Excitovaná/měřená frekvence	↓1e-10 ↑1e+10 ⊙1.0	Double (F64)
ad	Amplituda excitace	⊙0.01	Double (F64)

nper	Velikost okna (počet period fd)	↓1 ⊙4	Long (I32)
ifrunit	Jednotky pro frekvenci	⊙1	Long (I32)
	1 Hz		
	2 rad/s		
xi	Koeficient tlumení filtru	↓0.001 ↑100.0 ⊙1.0	Double (F64)
nmax	Rezervovaná paměť pro pole	↓10 ↑10000000 ⊙256	Long (I32)

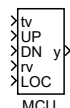
Výstup

ypv	Výstup řízené veličiny	Double (F64)
amp	Amplituda signálu po filtraci	Double (F64)
Ip	Výkonnost regulační smyčky	Double (F64)
E	Příznak chyby	Bool
iE	Kód chyby	Error

MCU – Jednotka pro ruční zadávání

Symbol bloku

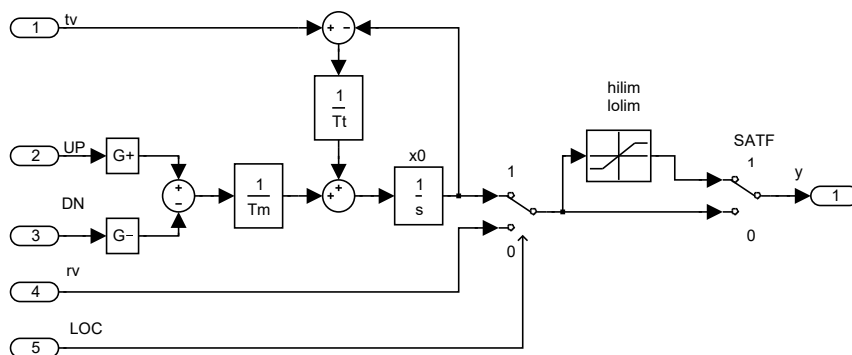
Licence: [STANDARD](#)



Popis funkce

Blok MCU je vhodný pro ruční nastavení číselné hodnoty výstupu y , například referenčního bodu. V lokálním režimu ($LOC = on$) je hodnota nastavena pomocí tlačítek UP a DN. Rychlost zvyšování/snížení výstupu y od počáteční hodnoty y_0 je určena časovou konstantou integrace t_m a dobou stisku tlačítek. Po uplynutí t_a sekund, kdy je tlačítko stisknuto, je rychlost vždy vynásobena faktorem q dokud neuplyne čas t_f . Volitelně může být rozsah výstupu y omezen ($SATF = on$) limitními hodnotami saturace $lolim$ a $hilim$. Pokud není stisknuto žádné tlačítko ($UP = off$ a $DN = off$), výstup y sleduje vstupní hodnotu tv . Rychlost sledování je kontrolována časovou konstantou integrace t_t .

V dálkovém režimu ($LOC = off$) je vstup rv volitelně saturován ($SATF = on$) a zkopírován na výstup y . Detailní funkce bloku je znázorněna na následujícím diagramu.



Tento blok propaguje kvalitu signálu. Více informací je uvedeno v sekci [1.4](#).

Vstup

tv	Veličina pro vysledování	Double (F64)
UP	Signál UP (nahoru, více)	Bool
DN	Signál DN (dolů, méně)	Bool
rv	Externí hodnota pro vzdálený režim	Double (F64)
LOC	Lokální nebo vzdálený režim	Bool

Parametr

<code>tt</code>	Časová konstanta vysledování	⊙1.0	Double (F64)
<code>tm</code>	Počáteční časová konstanta strmosti najíždění	⊙100.0	Double (F64)
<code>y0</code>	Počáteční hodnota výstupu		Double (F64)
<code>q</code>	Faktor určující velikost změny strmosti najíždění	⊙5.0	Double (F64)
<code>ta</code>	Interval, po kterém dochází ke zvýšení strmosti [s]	⊙4.0	Double (F64)
<code>tf</code>	Interval, po kterém se strmost již dále nemění [s]	⊙8.0	Double (F64)
<code>SATF</code>	Saturace		Bool
	<code>off</code> ... signál není omezen		
	<code>on</code> saturační meze jsou aktivní		
<code>hilim</code>	Horní saturační mez	⊙1.0	Double (F64)
<code>lolim</code>	Dolní saturační mez	⊙-1.0	Double (F64)

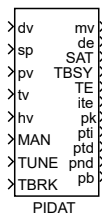
Výstup

<code>y</code>	Analogový výstupní signál	Double (F64)

PIDAT – PID regulátor s reléovým autotunerem

Symbol bloku

Licence: [AUTOTUNING](#)



Popis funkce

Blok **PIDAT** má stejnou regulační funkci jako blok **PIDU**. Kromě toho je vybaven funkcí automatického ladění pomocí relé.

Pro využití této funkce je nutné převést řízený systém do přibližně ustáleného stavu (ve vhodném pracovním bodě), zvolit požadovaný typ regulátoru (PI nebo PID) a aktivovat vstup **TUNE** hodnotou **on** (start identifikačního experimentu). V následném identifikačním experimentu je řízený proces regulován pomocí speciálního adaptivního reléového regulátoru a ze získaného záznamu vstupu a výstupu procesu je odhadnut vhodný bod jeho frekvenční charakteristiky. Na základě toho jsou poté určeny parametry regulátoru. Amplitudu reléového regulátoru (úroveň vybuzení systému) je možné nastavit parametrem **amp** a jeho hysterezi parametrem **hys**. Zvolíme-li **hys=0**, potom se hystereze relé určí automaticky na základě odhadu úrovně šumu měření regulované veličiny. Během identifikačního experimentu je **TBSY = on**. Během běžícího ladícího experimentu výstup **ite** ukazuje odhadovaný čas do konce v sekundách.

Po řádném skončení experimentu je **TE = off** a vypočítané parametry se objeví na výstupech **pk**, **pti**, **ptd**, **pnd**, **pb**. Váhový koeficient **c** je uvažován **c=0**. Skončil-li experiment s chybou, je **TE = on** a **ite** blíže specifikuje důvod chyby. Při výskytu chyby se doporučuje zvětšit parametr **amp**. Jeho volbu usnadňuje zabudovaná funkce, která parametr **amp** automaticky zmenšuje při hrozbě překročení maximální dovolené odchylky **maxdev** regulované veličiny od jejího počátečního ustáleného stavu. Identifikační experiment je možné předčasně ukončit aktivací vstupu **TBRK**.

Pro zesílení platí, že hodnota $K = 0$ vypne regulátor. Záporné hodnoty nejsou dovoleny (k tomu slouží parametr **RACT**). Nastavení časové integrační konstanty na $T_i = 0$ vypne integrační složku regulátoru (stejný efekt jako vypnutí parametrem **irtype**). Pro $T_d = 0$ dojde k vypnutí derivační složky regulátoru. Pro **iainf = 3,4,5** je nutné zadat statické zesílení procesu **k0**.

Je doporučeno neměnit parametry **n1**, **mm**, **ntime**, **rerrap** and **aerrph**.

Tento blok propaguje kvalitu signálu. Více informací je uvedeno v sekci 1.4.

Vstup

dv	Proměnná dopředné vazby	Double (F64)
sp	Požadovaná hodnota (setpoint)	Double (F64)
pv	Řízená veličina	Double (F64)
tv	Veličina pro vysledování	Double (F64)
hv	Hodnota výstupu v manuálním režimu	Double (F64)
MAN	Manuální nebo automatický režim	Bool
	off ... automatický režim	
	on manuální režim	
TUNE	Zahájení ladicího experimentu	Bool
TBRK	Ukončení ladicího experimentu	Bool

Parametr

irtype	Typ regulátoru	⊙6	Long (I32)
	1 [D]		
	2 [I]		
	3 [ID]		
	4 [P]		
	5 [PD]		
	6 PI		
	7 PID		
RACT	Převrácené působení výstupu regulátoru		Bool
	off ... vyšší mv -> vyšší pv		
	on vyšší mv -> nižší pv		
k	Zesílení regulátoru	↓0.0 ⊙1.0	Double (F64)
ti	Integrační časová konstanta	↓0.0 ⊙4.0	Double (F64)
td	Derivační časová konstanta	↓0.0 ⊙1.0	Double (F64)
nd	Parametr filtru derivační složky	↓0.0 ⊙10.0	Double (F64)
b	Váhový faktor pro proporcionální složku	↓0.0 ⊙1.0	Double (F64)
c	Váhový faktor pro derivační složku	↓0.0	Double (F64)
tt	Časová konstanta vysledování	↓0.0 ⊙1.0	Double (F64)
hilim	Horní mez akčního zásahu regulátoru	⊙1.0	Double (F64)
lolim	Dolní mez akčního zásahu regulátoru	⊙-1.0	Double (F64)
iainf	Druh apriorní informace	⊙1	Long (I32)
	1 žádná apriorní informace		
	2 astatický proces		
	3 proces nízkého řádu		
	4 statický proces + aperiodická odezva uzavřené smyčky		
	5 statický proces + středně rychlá odezva uzavřené smyčky		
	6 statický proces + rychlá odezva uzavřené smyčky		
k0	Statické zesílení	⊙1.0	Double (F64)
n1	Maximální počet púlperiod - bod frekv. char.	⊙20	Long (I32)

mm	Maximální počet púlperiod - průměrování	⊙4	Long (I32)
amp	Amplituda reléového regulátoru	⊙0.1	Double (F64)
uhys	Hystereze reléového regulátoru		Double (F64)
ntime	Doba odhadování šumu [s]	⊙5.0	Double (F64)
rerrap	Ukončovací hodnota - relativní chyba amplitudy	⊙0.1	Double (F64)
aerrph	Ukončovací hodnota - absolutní chyba fáze	⊙10.0	Double (F64)
maxdev	Maximální přípustná odchylka regulované veličiny	⊙1.0	Double (F64)

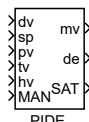
Výstup

mv	Akční zásah regulátoru (manipulated variable)	Double (F64)
de	Regulační odchylka	Double (F64)
SAT	Saturace	Bool
	off ... lineární zákon řízení	
	on výstup regulátoru je saturován	
TBSY	Příznak probíhajícího ladicího experimentu	Bool
TE	Příznak chyby během ladění	Bool
	off ... ladění proběhlo bez chyby	
	on během ladění se vyskytla chyba	
ite	Kód chyby	Long (I32)
	1000 .. příliš nízký poměr užitečného signálu k šumu měření	
	1001 .. příliš velká hystereze reléového regulátoru	
	1002 .. příliš přísné pravidlo pro ukončení	
	1003 .. příliš velká chyba při určování fáze identifikovaného bodu	
pk	Navržené zesílení regulátoru	Double (F64)
pti	Navržená integrační časová konstanta regulátoru	Double (F64)
ptd	Navržená derivační časová konstanta regulátoru	Double (F64)
pnd	Navržený parametr filtru derivační složky	Double (F64)
pb	Navržený váhový faktor pro proporcionální složku	Double (F64)

PIDE – PID regulátor se statikou

Symbol bloku

Licence: [ADVANCED](#)



Popis funkce

Blok **PIDE** je základní blok pro vytvoření úplného modifikovaného regulátoru PI(D), který se liší od standardního PI(D) regulátoru (blok **PIDU**) tím, že má zadané konečné statické zesílení (ve skutečnosti se zadává velikost odchylky ε , která způsobí saturaci výstupu). V nejjednodušším případě může pracovat zcela samostatně a plnit standardní funkci modifikovaného PID regulátoru s dvěma stupni volnosti v automatickém (**MAN = off**) nebo manuálním režimu (**MAN = on**).

V automatickém režimu v lineární oblasti realizuje zákon řízení daný vztahem

$$U(s) = \pm K \left[bW(s) - Y(s) + \frac{1}{T_i s + \beta} E(s) + \frac{T_d s}{\frac{T_d s}{N} + 1} (cW(s) - Y(s)) \right] + Z(s),$$

kde

$$\beta = \frac{K\varepsilon}{1 - K\varepsilon}$$

a $U(s)$ je obraz akční veličiny **mv**, $W(s)$ je obraz požadované hodnoty **sp**, $Y(s)$ je obraz regulované veličiny **pv**, $E(s)$ je Laplaceova transformace regulační odchylky, $Z(s)$ je obraz dopředné vazby **dv** a $K, T_i, T_d, N, \varepsilon (= b_p/100), b, c$ jsou parametry regulátoru. Znaménko pravé strany závisí na parametru **RACT**. Rozsah řídicí veličiny **mv** je omezen saturačními mezemi **lolim** a **hilim**.

Propojením výstupu **mv** se vstupem **tv** a vhodnou volbou parametru **tt** dosáhneme žádaného chování regulátoru při dosažení saturačních hodnot **mv**. Odstraníme tak nežádoucí unášení integrační složky (wind up effect) a současně s tím zajistíme bezrázové přepínání (bumpless transfer) automatického a manuálního režimu.

V manuálním režimu (**MAN = on**) je vstup **hv** (po případném omezení) kopírován na výstup **mv**. Signál připojený na vstup **tv** zajišťuje v tomto režimu příslušné vysledování vnitřního stavu regulátoru pro následné bezrázové přepnutí do automatického režimu (pro $\varepsilon > 0$ však vysledování není zcela přesné).

Tento blok propaguje kvalitu signálu. Více informací je uvedeno v sekci [1.4](#).

Vstup

dv

Proměnná dopředné vazby

Double (F64)

sp	Požadovaná hodnota (setpoint)	Double (F64)
pv	Řízená veličina	Double (F64)
tv	Veličina pro vysledování	Double (F64)
hv	Hodnota výstupu v manuálním režimu	Double (F64)
MAN	Manuální nebo automatický režim	Bool
	off ... automatický režim	
	on manuální režim	

Parametr

irtype	Typ regulátoru	⊙6	Long (I32)
	1 D		
	2 I		
	3 ID		
	4 P		
	5 PD		
	6 PI		
	7 PID		
RACT	Převrácené působení výstupu regulátoru		Bool
	off ... vyšší mv -> vyšší pv		
	on vyšší mv -> nižší pv		
k	Zesílení regulátoru	↓0.0 ⊙1.0	Double (F64)
ti	Integrační časová konstanta	↓0.0 ⊙4.0	Double (F64)
td	Derivační časová konstanta	↓0.0 ⊙1.0	Double (F64)
nd	Parametr filtru derivační složky	↓0.0 ⊙10.0	Double (F64)
b	Váhový faktor pro proporcionální složku	↓0.0 ⊙1.0	Double (F64)
c	Váhový faktor pro derivační složku	↓0.0	Double (F64)
tt	Časová konstanta vysledování	↓0.0 ⊙1.0	Double (F64)
bp	Koeficient chyby v ustáleném stavu		Double (F64)
hilim	Horní mez akčního zásahu regulátoru	⊙1.0	Double (F64)
lolim	Dolní mez akčního zásahu regulátoru	⊙-1.0	Double (F64)

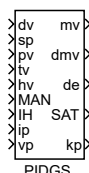
Výstup

mv	Akční zásah regulátoru (manipulated variable)	Double (F64)
de	Regulační odchylka	Double (F64)
SAT	Saturace	Bool
	off ... lineární zákon řízení	
	on výstup regulátoru je saturován	

PIDGS – PID regulátor s přepínáním sad parametrů

Symbol bloku

Licence: [ADVANCED](#)



Popis funkce

Regulační funkce bloku **PIDGS** je přesně shodná s blokem **PIDU**. Blok **PIDGS** má však až šest sad základních parametrů, které je možné bezrázově přepínat pomocí vstupu **ip** (index sady parametrů) nebo vstupu **vp** (přepínací analogová veličina). V případě použití přepínací analogové veličiny je třeba zadat **GSCF = on** a vektor příslušných přepínacích mezí **thrsha**. Sady parametrů jsou poté přepínány takto: sada 0 je pro $vp < thrsha(0)$, sada 1 pro $thrsha(0) < vp < thrsha(1)$ atd. až sada 5 pro $thrsha(4) < vp$. Index aktuální sady je k dispozici na výstupu **kp**.

Tento blok propaguje kvalitu signálu. Více informací je uvedeno v sekci [1.4](#).

Vstup

dv	Proměnná dopředné vazby	Double (F64)
sp	Požadovaná hodnota (setpoint)	Double (F64)
pv	Řízená veličina	Double (F64)
tv	Veličina pro vysledování	Double (F64)
hv	Hodnota výstupu v manuálním režimu	Double (F64)
MAN	Manuální nebo automatický režim off ... automatický režim on manuální režim	Bool
IH	Zastavení integrace off ... integrování povoleno on integrování pozastaveno	Bool
ip	Index sady parametrů	↓0 ↑5 Long (I32)
vp	Přepínací veličina	Double (F64)

Parametr

hilim	Horní mez akčního zásahu regulátoru	⊙1.0 Double (F64)
lolim	Dolní mez akčního zásahu regulátoru	⊙-1.0 Double (F64)
dz	Pásmo necitlivosti	Double (F64)

icotype	Typ výstupu regulátoru	⊙1	Long (I32)
	1 analogový		
	2 PWM		
	3 SCU		
	4 SCUV		
nmax	Rezervovaná paměť pro pole	↓4 ↑10000 ⊙10	Long (I32)
GSCF	Přepínání parametrů podle vstupu vp		Bool
	off ... přepínání indexem sady parametrů		
	on přepínání analogovým signálem		
hys	Hystereze pro přepínání podle vstupu vp		Double (F64)
irtypea	Vektor typů regulátoru	⊙[6 6 6 6 6 6]	Byte (U8)
RACTA	Vektor příznaků obráceného působení výstupu regulátoru	⊙[0 0 0 0 0 0]	Bool
ka	Vektor zesílení regulátoru	⊙[1.0 1.0 1.0 1.0 1.0 1.0]	Double (F64)
tia	Vektor integračních časových konstant	⊙[4.0 4.0 4.0 4.0 4.0 4.0]	Double (F64)
tda	Vektor derivačních časových konstant	⊙[1.0 1.0 1.0 1.0 1.0 1.0]	Double (F64)
nda	Vektor parametrů filtru derivační složky	⊙[10.0 10.0 10.0 10.0 10.0 10.0]	Double (F64)
ba	Váhové faktory pro proporcionální složku	⊙[1.0 1.0 1.0 1.0 1.0 1.0]	Double (F64)
ca	Váhové faktory pro derivační složku	⊙[0.0 0.0 0.0 0.0 0.0 0.0]	Double (F64)
tta	Vektor časových konstant výsledování	⊙[1.0 1.0 1.0 1.0 1.0 1.0]	Double (F64)
thrsha	Vektor mezí přepínací veličiny	⊙[0.1 0.2 0.3 0.4 0.5 0]	Double (F64)

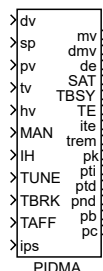
Výstup

mv	Akční zásah regulátoru (manipulated variable)	Double (F64)
dmv	Rychlostní výstup regulátoru (difference)	Double (F64)
de	Regulační odchylka	Double (F64)
SAT	Saturace	Bool
	off ... lineární zákon řízení	
	on výstup regulátoru je saturován	
kp	Index aktuální sady parametrů	Long (I32)

PIDMA – PID regulátor s momentovým autotunerem

Symbol bloku

Licence: [AUTOTUNING](#)



Popis funkce

Blok **PIDMA** má stejnou regulační funkci jako blok [PIDU](#). Kromě toho je vybaven funkcí automatického ladění pomocí momentového autotuneru.

V automatickém režimu (**MAN** = **off**) realizuje blok **PIDMA** řídicí zákon PID regulátoru se dvěma stupni volnosti ve tvaru:

$$U(s) = \pm K \left\{ bW(s) - Y(s) + \frac{1}{T_i s} [W(s) - Y(s)] + \frac{T_d s}{\frac{T_d}{N}s + 1} [cW(s) - Y(s)] \right\} + Z(s)$$

kde $U(s)$ je Laplaceova transformace řídicí veličiny **mv**, $W(s)$ je Laplaceova transformace požadované hodnoty **sp**, $Y(s)$ je Laplaceova transformace regulované veličiny **pv**, $Z(s)$ je Laplaceova transformace dopředné vazby **dv** a K , T_i , T_d , N , b , c jsou parametry regulátoru. Znaménko pravé strany závisí na parametru **RACT**. Rozsah řídicí veličiny **mv** (polohového výstupu regulátoru) je omezen parametry **hilim**, **lolim**. Parametr **dz** udává pásmo necitlivosti v integrační složce regulátoru. Navíc integrační složka může být vypnuta a zafixována na své aktuální hodnotě vstupem **IH** (**IH** = **on**). Pro správnou funkci regulátoru je nutné propojit výstup regulátoru **mv** se vstupem **tv** a správně nastavit časovou konstantu vysledování **tt**.

Doporučené nastavení časové konstanty **tt** pro PID regulátor je $\mathbf{tt} \approx \sqrt{T_i T_d}$ a pro PI regulátor $\mathbf{tt} \approx T_i/2$. Toto nastavení zajišťuje bezrázový přechod při přepínání režimu regulátoru (manuální, automatický) a správnou funkci regulátoru při saturaci výstupu **mv** (tzv. antiwindup). Úpravou parametru **tt** je v případě potřeby možné nastavit přesné chování v saturaci (tzv. odskakování od saturace vlivem šumu) a při přepínání více regulátorů (velikost skoku při přepnutí, pokud není nulová regulační odchylka). Hodnota 0 automaticky aplikuje doporučené nastavení pro PI a PID regulátory. U regulátorů bez integrační složky znamená deaktivaci sledování. Pro aktivaci sledování u P nebo PD regulátorů (např. pro regulaci kolem rovnovážného bodu) je třeba nastavit kladnou hodnotu **tt** vyšší než je vzorkovací perioda. Deaktivace sledování u regulátorů s integrační složkou není možná kvůli riziku windup efektu.

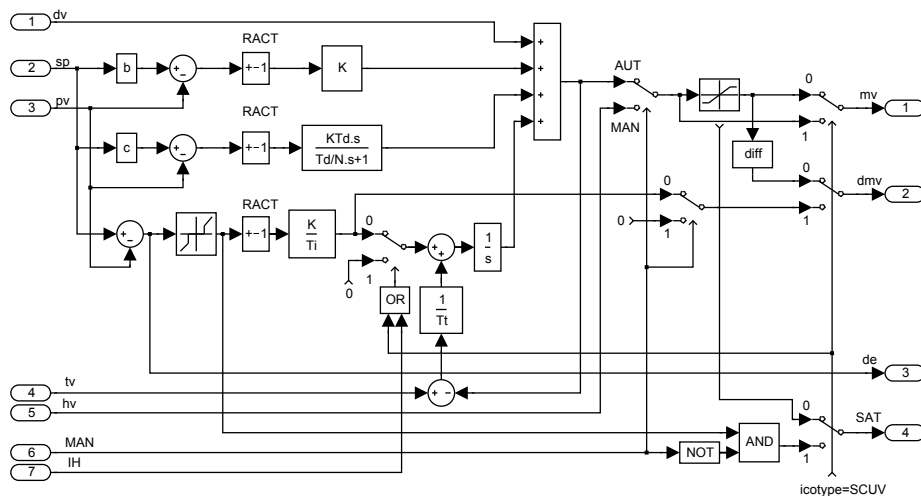
Pro zesílení platí, že hodnota $K = 0$ vypne regulátor. Záporné hodnoty nejsou dovoleny (k tomu slouží parametr **RACT**). Nastavení časové integrační konstanty na $T_i = 0$ vypne integrační složku regulátoru (stejný efekt jako vypnutí parametrem **irtype**). Pro $T_d = 0$ dojde k vypnutí derivační složky regulátoru. Přídavné výstupy **dmv**, **de** a **SAT** poskytují po řadě rychlostní výstup regulátoru (diference **mv**), regulační odchylku a příznak saturace výstupu regulátoru **mv**.

Bloku **PIDMA** je možné pomocí parametru **icotype** připravit na propojení s dalšími typy řídicích bloků. Parametr **icotype** může mít nastaveny následující hodnoty s danými významy:

- **1: analogový** - klasický režim bloku,
- **2: PWM** - režim vhodný pro zapojení výstupu **mv** na vstup pulzně-šířkově modulované regulace **PWM**,
- **3: SCU** - režim pro zapojení s krokovým regulátorem s polohovou zpětnou vazbou **SCU**,
- **4: SCUV** - režim pro zapojení s krokovým regulátorem bez polohové zpětné vazby **SCUV**.

Pro poslední možnost platí, že význam výstupů **mv**, **dmv** a **SAT** je v tomto případě pozměněn: výstup **mv** je roven součtu P a D složky regulátoru, zatímco výstup **dmv** poskytuje diferenci jeho I složky a výstup **SAT** nese informaci pro blok **SCUV**, zda je regulační odchylka **de** v automatickém režimu menší než pásmo necitlivosti **dz**. Pro případ propojení bloků **PIDMA** a **SCUV** se navíc doporučuje volit váhový koeficient požadované hodnoty pro derivační složku **c** rovný nule.

V manuálním režimu (**MAN = on**) je vstup **hv** kopírován na výstup **mv**, pokud nenaráží na horní či dolní omezení výstupu regulátoru. Celková regulační funkce bloku **PIDMA** je zřejmá z následujícího obrázku.

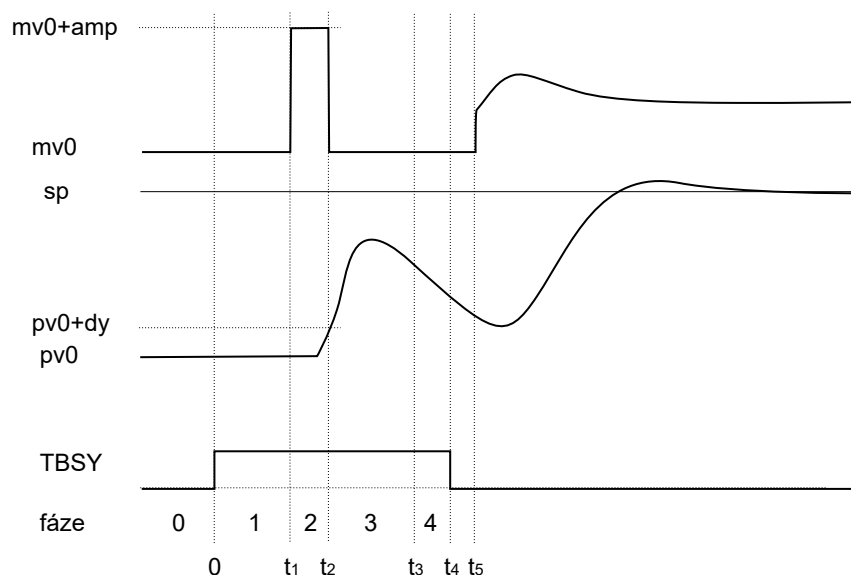


Blok **PIDMA** rozšiřuje řídicí funkci standardního PID regulátoru o vestavěné automatické nastavování parametrů (PID autotuner). Před spuštěním autotuneru musí operátor ve vhodném pracovním bodě dosáhnout ustáleného stavu a zvolit požadovaný typ regulátoru **ittype** (PI nebo PID) a nastavit další parametry autotuneru (**iainf**, **DGC**, **tdg**, **tn**, **amp**, **dy** a **ispeed**). Identifikační experiment se startuje náběžnou hranou **off**→**on** na vstupu **TUNE** (vstupem **TBRK** jej lze předčasně ukončit). V tomto módu (**TBSY** = **on**) je nejprve odhadnut drift a šum regulované veličiny (ve specifikovaném čase **tdg**+**tn**) a poté je na vstup procesu aplikován pravoúhlý puls. Z odezvy procesu jsou odhadnuty první tři momenty jeho impulsní odezvy. Amplituda pulsu se nastavuje parametrem **amp**. Puls je ukončen poté, co se hodnota regulované veličiny **pv** změní o více, než určuje tolerance (práh) **dy** (zadáva se vždy jako kladné číslo). Pokud je nastaven příznak **DGC**, používá se při zpracování signálu speciální kompenzace trendu signálu. Odhad času zbývajících do konce procesu ladění je přiveden na výstup **trem**.

Pokud experiment skončí úspěšně (**TE** = **off**), pak se v závislosti na vstupu **ips** objeví na výstupech:

- **0:** Navržené parametry **pk**, **pti**, **ptd**, **pnd**, **pb**, **pc**.
- **1:** Momenty procesu: zesílení (**pk**), míra zpoždění soustavy (**pti**), míra délky odezvy soustavy (**ptd**).
- **2:** Tříparametrový model procesu prvního řádu s dopravním zpožděním: zesílení (**pk**), dopravní zpoždění (**pti**), časová konstanta (**ptd**). Viz blok **FOPDT**.
- **3:** Tříparametrový model procesu druhého řádu s násobnou časovou konstantou a dopravním zpožděním: zesílení (**pk**), dopravní zpoždění (**pti**), časová konstanta (**ptd**). Viz blok **SOPDT**.
- **4:** Odhad mezí intervalu pro manuální doladění zesílení **k** PID regulátoru (**irtype** = 7): horní mez **k_{hi}** (**pk**), dolní mez **k_{lo}** (**pti**).

Další hodnoty vstupu **ips** jsou rezervovány pro speciální účely. Pro (**TE** = **on**) určuje výstup **ite** kód chyby experimentu. Funkce autotuneru je demonstrována na následujícím obrázku.



Během identifikačního experimentu výstup `ite` indikuje jednotlivé fáze činnosti autotuneru. Ve fázi odhadu strmosti odeznívání odezvy (`ite = -4`) může být proces ladění předčasně manuálně ukončen. V tomto případě jsou parametry regulátoru řádně navrženy, avšak jejich možná nepřesnost je indikována varovným kódem `ite = 100`.

Poznámka: Náběžná hrana `off`→`on` na vstupu `TUNE` během fází -2, -3 a -4 způsobí ukončení aktuální fáze a přechod na další (nebo dokončení experimentu ve fázi -4).

Po ukončení experimentu (`TBSY on`→`off`) je funkce regulátoru závislá na nastaveném režimu (manuální, automatický). Jestliže `TAFF = on`, potom jsou navržené parametry okamžitě použity.

Tento blok propaguje kvalitu signálu. Více informací je uvedeno v sekci 1.4.

Vstup

<code>dv</code>	Proměnná dopředné vazby	Double (F64)
<code>sp</code>	Požadovaná hodnota (setpoint)	Double (F64)
<code>pv</code>	Řízená veličina	Double (F64)
<code>tv</code>	Veličina pro vysledování	Double (F64)
<code>hv</code>	Hodnota výstupu v manuálním režimu	Double (F64)
<code>MAN</code>	Manuální nebo automatický režim	Bool
	<code>off</code> ... automatický režim	
	<code>on</code> manuální režim	
<code>IH</code>	Zastavení integrace	Bool
	<code>off</code> ... integrování povoleno	
	<code>on</code> integrování pozastaveno	
<code>TUNE</code>	Zahájení ladicího experimentu	Bool
<code>TBRK</code>	Ukončení ladicího experimentu	Bool

TAFF	Přijetí výsledků ladícího experimentu	Bool
	off ... parametry jsou pouze vypočítány	
	on ... parametry jsou dosazeny do řídicího algoritmu	
ips	Význam výstupních signálů	Long (I32)
	0 parametry PID regulátoru	
	1 momenty procesu	
	2 model prvního řádu s dopravním zpožděním	
	3 model druhého řádu s násobnou časovou konstantou a dopravním zpožděním	
	4 meze pro manuální ladění zesílení regulátoru	

Parametr

irtype	Typ regulátoru	⊙6	Long (I32)
	1 D		
	2 I		
	3 ID		
	4 P		
	5 PD		
	6 PI		
	7 PID		
RACT	Převrácené působení výstupu regulátoru		Bool
	off ... vyšší mv -> vyšší pv		
	on ... vyšší mv -> nižší pv		
k	Zesílení regulátoru	↓0.0 ⊙1.0	Double (F64)
ti	Integrační časová konstanta	↓0.0 ⊙4.0	Double (F64)
td	Derivační časová konstanta	↓0.0 ⊙1.0	Double (F64)
nd	Parametr filtru derivační složky	↓0.0 ⊙10.0	Double (F64)
b	Váhový faktor pro proporcionální složku	↓0.0 ↑2.0 ⊙1.0	Double (F64)
c	Váhový faktor pro derivační složku	↓0.0 ↑2.0	Double (F64)
tt	Časová konstanta výsledování	↓0.0 ⊙1.0	Double (F64)
hilim	Horní mez akčního zásahu regulátoru	⊙1.0	Double (F64)
lolim	Dolní mez akčního zásahu regulátoru	⊙-1.0	Double (F64)
dz	Pásmo necitlivosti		Double (F64)
icotype	Typ výstupu regulátoru	⊙1	Long (I32)
	1 analogový		
	2 PWM		
	3 SCU		
	4 SCUV		
itttype	Požadovaný typ regulátoru pro návrh	⊙6	Long (I32)
	6 PI		
	7 PID		
iainf	Druh apriorní informace	⊙1	Long (I32)
	1 statický proces		
	2 astatický proces		

DGC	Kompenzace driftu gradientu	⊙on	Bool
	off ... zakázáno		
	on povoleno		
tdg	Doba odhadu gradientu trendu [s]	⊙60.0	Double (F64)
tn	Doba odhadování šumu [s]	⊙5.0	Double (F64)
amp	Amplituda ladícího pulzu	⊙0.5	Double (F64)
dy	Práh pro ukončení pulsu	↓0.0 ⊙0.1	Double (F64)
ispeed	Požadovaná rychlost uzavřené smyčky	⊙2	Long (I32)
	1 pomalá uzavřená smyčka		
	2 středně rychlá uzavřená smyčka		
	3 rychlá uzavřená smyčka		
ipid	Forma PID regulátoru	⊙1	Long (I32)
	1 paralelní realizace		
	2 sériová realizace		

Výstup

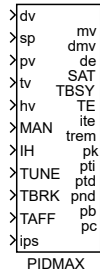
mv	Akční zásah regulátoru (manipulated variable)	Double (F64)
dmv	Rychlostní výstup regulátoru (difference)	Double (F64)
de	Regulační odchylka	Double (F64)
SAT	Saturace	Bool
	off ... lineární zákon řízení	
	on výstup regulátoru je saturován	
TBSY	Příznak probíhajícího ladícího experimentu	Bool
TE	Příznak chyby během ladění	Bool
	off ... ladění proběhlo bez chyby	
	on během ladění se vyskytla chyba	

ite	Kód chyby	Long (I32)
	0 bez chyby	
	1 příliš malá hodnota prahu pro ukončení pulzu	
	2 příliš velká amplituda pulzu	
	3 nebylo dosaženo ustáleného stavu	
	4 příliš malá amplituda pulzu	
	5 nebylo dosaženo ustáleného stavu	
	6 při experimentu došlo k saturaci výstupu regulátoru	
	7 pro vybraný typ regulátoru není podporováno automatické nastavování	
	8 nedodržena podmínka monotónnosti procesu	
	9 selhání extrapolace	
	10 neočekávané hodnoty momentů (fatální)	
	11 ruční přerušení experimentu uživatelem	
	12 nesprávný směr řídicí veličiny (změňte parametr RACT)	
	13 nesprávný formát vstupního řetězce sParams	
	100 ... ruční ukončení ladění (varování)	
	-1 odhad driftu a šumu (parametry t _{dg} a t _n)	
	-2 generování obdélníkového pulzu (pulz končí při změně p _v o hodnotu větší než d _y)	
	-3 hledání vrcholu odezvy	
	-4 odhad rychlosti ustalování odezvy	
trem	Odhad času do ukončení experimentu [s]	Double (F64)
pk	Navržené zesílení regulátoru	Double (F64)
pti	Navržená integrační časová konstanta regulátoru	Double (F64)
ptd	Navržená derivační časová konstanta regulátoru	Double (F64)
pnd	Navržený parametr filtru derivační složky	Double (F64)
pb	Navržený váhový faktor pro proporcionální složku	Double (F64)
pc	Navržený váhový faktor pro derivační složku	Double (F64)

PIDMAX – PID regulátor s rozšířeným momentovým autotunem

Symbol bloku

Licence: [AUTOTUNING](#)



Popis funkce

Funkční blok **PIDMAX** (PIDMA eXtended) je novou, rozšířenou verzí bloku [PIDMA](#).

V automatickém režimu (**MAN** = **off**) realizuje blok **PIDMAX** řídicí zákon PID regulátoru se dvěma stupni volnosti ve tvaru:

$$U(s) = \pm K \left\{ bW(s) - Y(s) + \frac{1}{T_i s} [W(s) - Y(s)] + \frac{T_d s}{N s + 1} [cW(s) - Y(s)] \right\} + Z(s)$$

kde $U(s)$ je Laplaceova transformace řídicí veličiny **mv**, $W(s)$ je Laplaceova transformace požadované hodnoty **sp**, $Y(s)$ je Laplaceova transformace regulované veličiny **pv**, $Z(s)$ je Laplaceova transformace dopředné vazby **dv** a K , T_i , T_d , N , b , c jsou parametry regulátoru. Znaménko pravé strany závisí na parametru **RACT**. Rozsah řídicí veličiny **mv** (polohového výstupu regulátoru) je omezen parametry **hilim**, **lolim**. Parametr **dz** udává pásmo necitlivosti v integrační složce regulátoru. Navíc integrační složka může být vypnuta a zafixována na své aktuální hodnotě vstupem **IH** (**IH** = **on**). Pro správnou funkci regulátoru je nutné propojit výstup regulátoru **mv** se vstupem **tv** a správně nastavit časovou konstantu sledování **tt**.

Doporučené nastavení časové konstanty **tt** pro PID regulátor je $tt \approx \sqrt{T_i T_d}$ a pro PI regulátor $tt \approx T_i/2$. Toto nastavení zajišťuje bezrázový přechod při přepínání režimu regulátoru (manuální, automatický) a správnou funkci regulátoru při saturaci výstupu **mv** (tzv. antiwindup). Úpravou parametru **tt** je v případě potřeby možné nastavit přesné chování v saturaci (tzv. odskakování od saturace vlivem šumu) a při přepínání více regulátorů (velikost skoku při přepnutí, pokud není nulová regulační odchylka). Hodnota 0 automaticky aplikuje doporučené nastavení pro PI a PID regulátory. U regulátorů bez integrační složky znamená deaktivaci sledování. Pro aktivaci sledování u P nebo PD regulátorů (např. pro regulaci kolem rovnovážného bodu) je třeba nastavit kladnou hodnotu **tt** vyšší než je vzorkovací perioda. Deaktivace sledování u regulátorů s integrační složkou není možná kvůli riziku windup efektu.

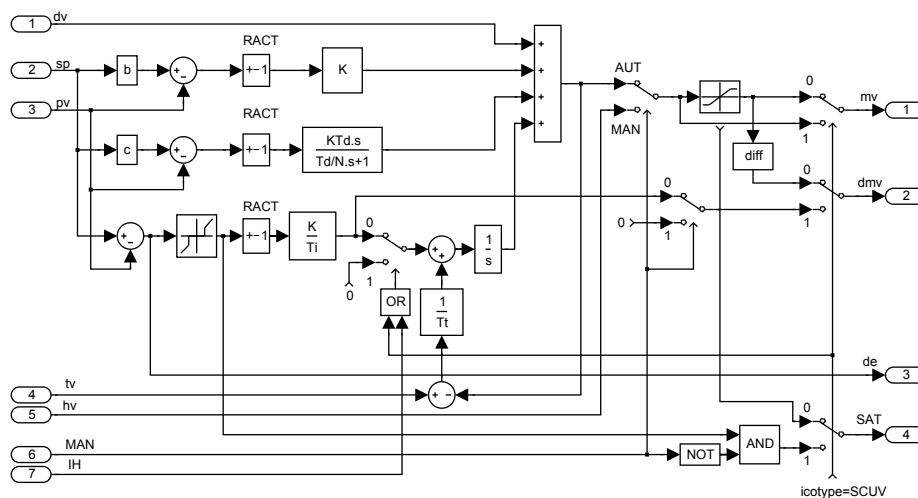
Pro zesílení platí, že hodnota $K = 0$ vypne regulátor. Záporné hodnoty nejsou dovoleny (k tomu slouží parametr **RACT**). Nastavení časové integrační konstanty na $T_i = 0$ vypne integrační složku regulátoru (stejný efekt jako vypnutí parametrem **irtype**). Pro $T_d = 0$ dojde k vypnutí derivační složky regulátoru. Přídavné výstupy **dmv**, **de** a **SAT** poskytují po řadě rychlostní výstup regulátoru (diference **mv**), regulační odchylku a příznak saturace výstupu regulátoru **mv**.

Bloku **PIDMAX** je možné pomocí parametru **icotype** připravit na propojení s dalšími typy řídicích bloků. Parametr **icotype** může mít nastaveny následující hodnoty s danými významy:

- **1: analogový** - klasický režim bloku,
- **2: PWM** - režim vhodný pro zapojení výstupu **mv** na vstup pulzně-šířkově modulované regulace **PWM**,
- **3: SCU** - režim pro zapojení s krokovým regulátorem s polohovou zpětnou vazbou **SCU**,
- **4: SCUV** - režim pro zapojení s krokovým regulátorem bez polohové zpětné vazby **SCUV**.

Pro poslední možnost platí, že význam výstupů **mv**, **dmv** a **SAT** je v tomto případě pozměněn: výstup **mv** je roven součtu P a D složky regulátoru, zatímco výstup **dmv** poskytuje diferenci jeho I složky a výstup **SAT** nese informaci pro blok **SCUV**, zda je regulační odchylka **de** v automatickém režimu menší než pásmo necitlivosti **dz**. Pro případ propojení bloků **PIDMAX** a **SCUV** se navíc doporučuje volit váhový koeficient požadované hodnoty pro derivační složku **c** rovný nule.

V manuálním režimu (**MAN = on**) je vstup **hv** kopírován na výstup **mv**, pokud nenarazí na horní či dolní omezení výstupu regulátoru. Celková regulační funkce bloku **PIDMAX** je zřejmá z následujícího obrázku.

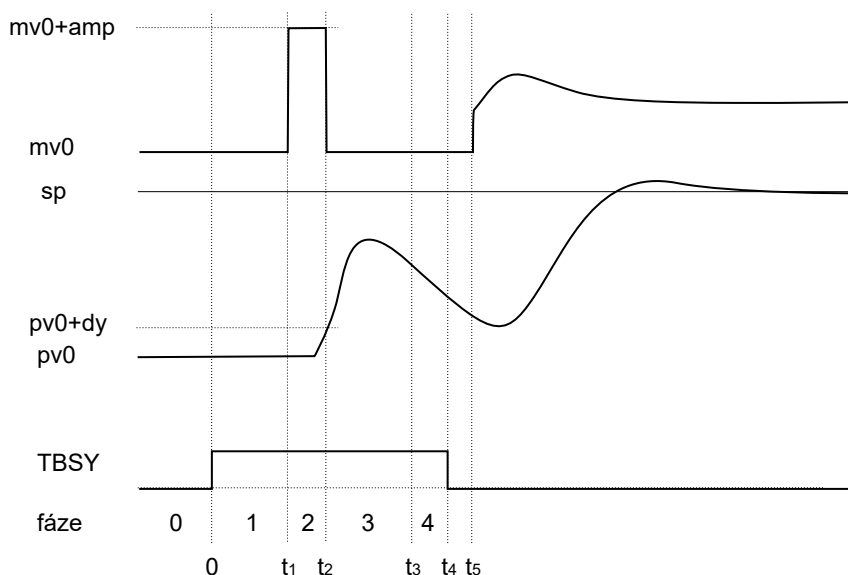


Blok **PIDMAX** rozšiřuje řídicí funkci standardního PID regulátoru o vestavěné automatické nastavování parametrů (PID autotuner). Před spuštěním autotuneru musí operátor ve vhodném pracovním bodě dosáhnout ustáleného stavu a zvolit požadovaný typ regulátoru **ittype** (PI nebo PID) a nastavit další parametry autotuneru (**iainf**, **DGC**, **tdg**, **tn**, **amp**, **dy**, **ispeed** a **iobj**). Identifikační experiment se startuje náběžnou hranou **off**→**on** na vstupu **TUNE** (vstupem **TBRK** jej lze předčasně ukončit). V tomto módu (**TBSY** = **on**) je nejprve odhadnut drift a šum regulované veličiny (ve specifikovaném čase **tdg**+**tn**) a poté je na vstup procesu aplikován pravoúhlý puls. Z odezvy procesu jsou odhadnuty první tři momenty jeho impulsní odezvy. Amplituda pulsu se nastavuje parametrem **amp**. Puls je ukončen poté, co se hodnota regulované veličiny **p**_v změní o více, než určuje tolerance (práh) **dy** (zadáva se vždy jako kladné číslo). Pokud je nastaven příznak **DGC**, používá se při zpracování signálu speciální kompenzace trendu signálu. Odhad času zbývajících do konce procesu ladění je přiveden na výstup **trem**.

Pokud experiment skončí úspěšně (**TE** = **off**), pak se v závislosti na vstupu **ips** objeví na výstupech:

- **0**: Navržené parametry **pk**, **pti**, **ptd**, **pnd**, **pb**, **pc**.
- **1**: Momenty procesu: zesílení (**pk**), míra zpoždění soustavy (**pti**), míra délky odezvy soustavy (**ptd**).
- **2**: Tříparametrový model procesu prvního řádu s dopravním zpožděním: zesílení (**pk**), dopravní zpoždění (**pti**), časová konstanta (**ptd**). Viz blok **FOPDT**.
- **3**: Tříparametrový model procesu druhého řádu s násobnou časovou konstantou a dopravním zpožděním: zesílení (**pk**), dopravní zpoždění (**pti**), časová konstanta (**ptd**). Viz blok **SOPDT**.
- **4**: Odhad mezí intervalu pro manuální doladění zesílení **k** PID regulátoru (**irtype** = 7): horní mez **k_{hi}** (**pk**), dolní mez **k_{lo}** (**pti**).

Další hodnoty vstupu **ips** jsou rezervovány pro speciální účely. Pro (**TE** = **on**) určuje výstup **ite** kód chyby experimentu. Funkce autotuneru je demonstrována na následujícím obrázku.



Během identifikačního experimentu výstup `ite` indikuje jednotlivé fáze činnosti autotuneru. Ve fázi odhadu strmosti odeznívání odezvy (`ite = -4`) může být proces ladění předčasně manuálně ukončen. V tomto případě jsou parametry regulátoru řádně navrženy, avšak jejich možná nepřesnost je indikována varovným kódem `ite = 100`.

Poznámka: Náběžná hrana `off`→`on` na vstupu `TUNE` během fází -2, -3 a -4 způsobí ukončení aktuální fáze a přechod na další (nebo dokončení experimentu ve fázi -4).

Po ukončení experimentu (`TBSY on`→`off`) je funkce regulátoru závislá na nastaveném režimu (manuální, automatický). Jestliže `TAFF = on`, potom jsou navržené parametry okamžitě použity.

Tento blok propaguje kvalitu signálu. Více informací je uvedeno v sekci 1.4.

Vstup

<code>dv</code>	Proměnná dopředné vazby	Double (F64)
<code>sp</code>	Požadovaná hodnota (setpoint)	Double (F64)
<code>pv</code>	Řízená veličina	Double (F64)
<code>tv</code>	Velichina pro vysledování	Double (F64)
<code>hv</code>	Hodnota výstupu v manuálním režimu	Double (F64)
<code>MAN</code>	Manuální nebo automatický režim	Bool
	<code>off</code> ... automatický režim	
	<code>on</code> ... manuální režim	
<code>IH</code>	Zastavení integrace	Bool
	<code>off</code> ... integrování povoleno	
	<code>on</code> ... integrování pozastaveno	
<code>TUNE</code>	Zahájení ladicího experimentu	Bool
<code>TBRK</code>	Ukončení ladicího experimentu	Bool

TAFF	Přijetí výsledků ladícího experimentu	Bool
	off ... parametry jsou pouze vypočítány	
	on ... parametry jsou dosazeny do řídicího algoritmu	
ips	Význam výstupních signálů	Long (I32)
	0 parametry PID regulátoru	
	1 momenty procesu	
	2 model prvního řádu s dopravním zpožděním	
	3 model druhého řádu s násobnou časovou konstantou a dopravním zpožděním	
	4 meze pro manuální ladění zesílení regulátoru	

Parametr

irtype	Typ regulátoru	⊙6	Long (I32)
	1 D		
	2 I		
	3 ID		
	4 P		
	5 PD		
	6 PI		
	7 PID		
RACT	Převrácené působení výstupu regulátoru		Bool
	off ... vyšší mv -> vyšší pv		
	on ... vyšší mv -> nižší pv		
k	Zesílení regulátoru	↓0.0 ⊙1.0	Double (F64)
ti	Integrační časová konstanta	↓0.0 ⊙4.0	Double (F64)
td	Derivační časová konstanta	↓0.0 ⊙1.0	Double (F64)
nd	Parametr filtru derivační složky	↓0.0 ⊙10.0	Double (F64)
b	Váhový faktor pro proporcionální složku	↓0.0 ↑2.0 ⊙1.0	Double (F64)
c	Váhový faktor pro derivační složku	↓0.0 ↑2.0	Double (F64)
tt	Časová konstanta výsledování	↓0.0 ⊙1.0	Double (F64)
hilim	Horní mez akčního zásahu regulátoru	⊙1.0	Double (F64)
lolim	Dolní mez akčního zásahu regulátoru	⊙-1.0	Double (F64)
dz	Pásmo necitlivosti		Double (F64)
icotype	Typ výstupu regulátoru	⊙1	Long (I32)
	1 analogový		
	2 PWM		
	3 SCU		
	4 SCUV		
itttype	Požadovaný typ regulátoru pro návrh	⊙6	Long (I32)
	6 PI		
	7 PID		
iainf	Druh apriorní informace	⊙1	Long (I32)
	1 statický proces		
	2 astatický proces		

DGC	Kompenzace driftu gradientu	⊙on	Bool
	off ... zakázáno		
	on povoleno		
tdg	Doba odhadu gradientu trendu [s]	⊙60.0	Double (F64)
tn	Doba odhadování šumu [s]	⊙5.0	Double (F64)
amp	Amplituda ladícího pulzu	⊙0.5	Double (F64)
dy	Práh pro ukončení pulsu	↓0.0 ⊙0.1	Double (F64)
ispeed	Požadovaná rychlost uzavřené smyčky	⊙2	Long (I32)
	1 pomalá uzavřená smyčka		
	2 středně rychlá uzavřená smyčka		
	3 rychlá uzavřená smyčka		
ipid	Forma PID regulátoru	⊙1	Long (I32)
	1 paralelní realizace		
	2 sériová realizace		
iobj	Cíl ladění	⊙3	Long (I32)
	0 Ekvivalentní s PIDMA		
	1 Sledování požadované hodnoty (servo)		
	2 Servo & regulátor		
	3 Potlačení poruchy (regulátor)		

Výstup

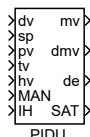
mv	Akční zásah regulátoru (manipulated variable)	Double (F64)
dmv	Rychlostní výstup regulátoru (diference)	Double (F64)
de	Regulační odchylka	Double (F64)
SAT	Saturace	Bool
	off ... lineární zákon řízení	
	on výstup regulátoru je saturován	
TBSY	Příznak probíhajícího ladícího experimentu	Bool
TE	Příznak chyby během ladění	Bool
	off ... ladění proběhlo bez chyby	
	on během ladění se vyskytla chyba	

ite	Kód chyby	Long (I32)
	0 bez chyby	
	1 příliš malá hodnota prahu pro ukončení pulzu	
	2 příliš velká amplituda pulzu	
	3 nebylo dosaženo ustáleného stavu	
	4 příliš malá amplituda pulzu	
	5 nebylo dosaženo ustáleného stavu	
	6 při experimentu došlo k saturaci výstupu regulátoru	
	7 pro vybraný typ regulátoru není podporováno automatické nastavování	
	8 nedodržena podmínka monotónnosti procesu	
	9 selhání extrapolace	
	10 neočekávané hodnoty momentů (fatální)	
	11 ruční přerušení experimentu uživatelem	
	12 nesprávný směr řídicí veličiny (změňte parametr RACT)	
	13 nesprávný formát vstupního řetězce sParams	
	100 ... ruční ukončení ladění (varování)	
	-1 odhad driftu a šumu (parametry t _{dg} a t _n)	
	-2 generování obdélníkového pulzu (pulz končí při změně p _v o hodnotu větší než d _y)	
	-3 hledání vrcholu odezvy	
	-4 odhad rychlosti ustalování odezvy	
trem	Odhad času do ukončení experimentu [s]	Double (F64)
pk	Navržené zesílení regulátoru	Double (F64)
pti	Navržená integrační časová konstanta regulátoru	Double (F64)
ptd	Navržená derivační časová konstanta regulátoru	Double (F64)
pnd	Navržený parametr filtru derivační složky	Double (F64)
pb	Navržený váhový faktor pro proporcionální složku	Double (F64)
pc	Navržený váhový faktor pro derivační složku	Double (F64)

PIDU – PID regulátor

Symbol bloku

Licence: [STANDARD](#)



Popis funkce

Blok PIDU představuje základní prvek pro realizaci kompletního PID regulátoru (P, I, PI, PD, PID, PI+S). V základním nastavení může fungovat samostatně jako standardní PID regulátor s dvěma stupni volnosti. Podporuje jak automatický (**MAN = off**), tak manuální režim (**MAN = on**).

V automatickém režimu (**MAN = off**) realizuje blok PIDU řídicí zákon PID regulátoru se dvěma stupni volnosti ve tvaru:

$$U(s) = \pm K \left\{ bW(s) - Y(s) + \frac{1}{T_i s} [W(s) - Y(s)] + \frac{T_d s}{\frac{T_d}{N}s + 1} [cW(s) - Y(s)] \right\} + Z(s)$$

kde $U(s)$ je Laplaceova transformace řídicí veličiny **mv**, $W(s)$ je Laplaceova transformace požadované hodnoty **sp**, $Y(s)$ je Laplaceova transformace regulované veličiny **pv**, $Z(s)$ je Laplaceova transformace dopředné vazby **dv** a K , T_i , T_d , N , b , c jsou parametry regulátoru. Znaménko pravé strany závisí na parametru **RACT**. Rozsah řídicí veličiny **mv** (polohového výstupu regulátoru) je omezen parametry **hilim**, **lolim**. Parametr **dz** udává pásmo necitlivosti v integrační složce regulátoru. Navíc integrační složka může být vypnuta a zafixována na své aktuální hodnotě vstupem **IH** (**IH = on**). Pro správnou funkci regulátoru je nutné propojit výstup regulátoru **mv** se vstupem **tv** a správně nastavit časovou konstantu výsledování **tt**.

Doporučené nastavení časové konstanty **tt** pro PID regulátor je $\mathbf{tt} \approx \sqrt{T_i T_d}$ a pro PI regulátor $\mathbf{tt} \approx T_i/2$. Toto nastavení zajišťuje bezrázový přechod při přepínání režimu regulátoru (manuální, automatický) a správnou funkci regulátoru při saturaci výstupu **mv** (tzv. antiwindup). Úpravou parametru **tt** je v případě potřeby možné nastavit přesné chování v saturaci (tzv. odskakování od saturace vlivem šumu) a při přepínání více regulátorů (velikost skoku při přepnutí, pokud není nulová regulační odchylka). Hodnota 0 automaticky aplikuje doporučené nastavení pro PI a PID regulátory. U regulátorů bez integrační složky znamená deaktivaci sledování. Pro aktivaci sledování u P nebo PD regulátorů (např. pro regulaci kolem rovnovážného bodu) je třeba nastavit kladnou hodnotu **tt** vyšší než je vzorkovací perioda. Deaktivace sledování u regulátorů s integrační složkou není možná kvůli riziku windup efektu.

Pro zesílení platí, že hodnota $K = 0$ vypne regulátor. Záporné hodnoty nejsou dovoleny (k tomu slouží parametr **RACT**). Nastavení časové integrační konstanty na $T_i = 0$

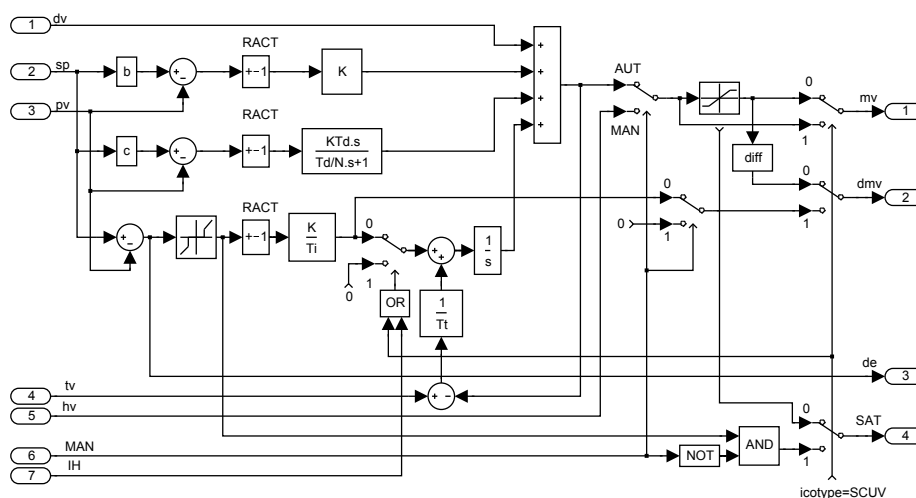
vypne integrační složku regulátoru (stejný efekt jako vypnutí parametrem `irtype`). Pro $T_d = 0$ dojde k vypnutí derivační složky regulátoru. Přídavné výstupy `dmv`, `de` a `SAT` poskytují po řadě rychlostní výstup regulátoru (diference `mv`), regulační odchylku a příznak saturace výstupu regulátoru `mv`.

Bloku `PIDU` je možné pomocí parametru `icotype` připravit na propojení s dalšími typy řídicích bloků. Parametr `icotype` může mít nastaveny následující hodnoty s danými významy:

- 1 **analogový** - klasický režim bloku,
- 2 **PWM** - režim vhodný pro zapojení výstupu `mv` na vstup pulzně-šířkově modulované regulace [PWM](#),
- 3 **SCU** - režim pro zapojení s krokovým regulátorem s polohovou zpětnou vazbou [SCU](#),
- 4 **SCUV** - režim pro zapojení s krokovým regulátorem bez polohové zpětné vazby [SCUV](#).

Pro poslední možnost platí, že význam výstupů `mv`, `dmv` a `SAT` je v tomto případě pozměněn: výstup `mv` je roven součtu P a D složky regulátoru, zatímco výstup `dmv` poskytuje diferenci jeho I složky a výstup `SAT` nese informaci pro blok [SCUV](#), zda je regulační odchylka `de` v automatickém režimu menší než pásmo necitlivosti `dz`. Pro případ propojení bloků `PIDU` a [SCUV](#) se navíc doporučuje volit váhový koeficient požadované hodnoty pro derivační složku `c` rovný nule.

V manuálním režimu (`MAN = on`) je vstup `hv` kopírován na výstup `mv`, pokud nenarazí na horní či dolní omezení výstupu regulátoru. Celková regulační funkce bloku `PIDU` je zřejmá z následujícího obrázku.



Tento blok propaguje kvalitu signálu. Více informací je uvedeno v sekci [1.4](#).

Vstup

dv	Proměnná dopředné vazby	Double (F64)
sp	Požadovaná hodnota (setpoint)	Double (F64)
pv	Řízená veličina	Double (F64)
tv	Veličina pro vysledování	Double (F64)
hv	Hodnota výstupu v manuálním režimu	Double (F64)
MAN	Manuální nebo automatický režim	Bool
	off ... automatický režim	
	on manuální režim	
IH	Zastavení integrace	Bool
	off ... integrování povoleno	
	on integrování pozastaveno	

Parametr

irtype	Typ regulátoru	⊙6	Long (I32)
	1 D		
	2 I		
	3 ID		
	4 P		
	5 PD		
	6 PI		
	7 PID		
RACT	Převrácené působení výstupu regulátoru		Bool
	off ... vyšší mv -> vyšší pv		
	on vyšší mv -> nižší pv		
k	Zesílení regulátoru	↓0.0 ⊙1.0	Double (F64)
ti	Integrační časová konstanta	↓0.0 ⊙4.0	Double (F64)
td	Derivační časová konstanta	↓0.0 ⊙1.0	Double (F64)
nd	Parametr filtru derivační složky	↓0.0 ⊙10.0	Double (F64)
b	Váhový faktor pro proporcionální složku	↓0.0 ↑2.0 ⊙1.0	Double (F64)
c	Váhový faktor pro derivační složku	↓0.0 ↑2.0	Double (F64)
tt	Časová konstanta vysledování	↓0.0 ⊙1.0	Double (F64)
hilim	Horní mez akčního zásahu regulátoru	⊙1.0	Double (F64)
lolim	Dolní mez akčního zásahu regulátoru	⊙-1.0	Double (F64)
dz	Pásmo necitlivosti		Double (F64)
icotype	Typ výstupu regulátoru	⊙1	Long (I32)
	1 analogový		
	2 PWM		
	3 SCU		
	4 SCUV		

Výstup

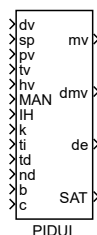
mv	Akční zásah regulátoru (manipulated variable)	Double (F64)

dmv	Rychlostní výstup regulátoru (diference)	Double (F64)
de	Regulační odchylka	Double (F64)
SAT	Saturace	Bool
	off ... lineární zákon řízení	
	on výstup regulátoru je saturován	

PIDUI – PID regulátor s parametry na vstupech

Symbol bloku

Licence: [ADVANCED](#)



Popis funkce

Regulační funkce bloku **PIDUI** je přesně shodná s blokem **PIDU**. Jediný rozdíl spočívá v tom, že základní parametry PID algoritmu jsou vyvedeny na vstupy. V důsledku toho je lze pohodlně měnit v závislosti na výstupech jiných bloků. Tímto způsobem lze realizovat speciální adaptivní PID regulátory.

Tento blok propaguje kvalitu signálu. Více informací je uvedeno v sekci [1.4](#).

Vstup

dv	Proměnná dopředné vazby	Double (F64)
sp	Požadovaná hodnota (setpoint)	Double (F64)
pv	Řízená veličina	Double (F64)
tv	Veličina pro vysledování	Double (F64)
hv	Hodnota výstupu v manuálním režimu	Double (F64)
MAN	Manuální nebo automatický režim	Bool
	off ... automatický režim	
	on manuální režim	
IH	Zastavení integrace	Bool
	off ... integrování povoleno	
	on integrování pozastaveno	
k	Zesílení regulátoru	Double (F64)
ti	Integrační časová konstanta	Double (F64)
td	Derivační časová konstanta	Double (F64)
nd	Parametr filtru derivační složky	Double (F64)
b	Váhový faktor pro proporcionální složku	Double (F64)
c	Váhový faktor pro derivační složku	Double (F64)

Parametr

<code>irtype</code>	Typ regulátoru	⊙6	Long (I32)
	1 D		
	2 I		
	3 ID		
	4 P		
	5 PD		
	6 PI		
	7 PID		
<code>RACT</code>	Převrácené působení výstupu regulátoru		Bool
	<code>off</code> ... vyšší mv -> vyšší pv		
	<code>on</code> ... vyšší mv -> nižší pv		
<code>tt</code>	Časová konstanta vysledování	⊙1.0	Double (F64)
<code>hilim</code>	Horní mez akčního zásahu regulátoru	⊙1.0	Double (F64)
<code>lolim</code>	Dolní mez akčního zásahu regulátoru	⊙-1.0	Double (F64)
<code>dz</code>	Pásmo necitlivosti		Double (F64)
<code>icotype</code>	Typ výstupu regulátoru	⊙1	Long (I32)
	1 analogový		
	2 PWM		
	3 SCU		
	4 SCUV		

Výstup

mv	Akční zásah regulátoru (manipulated variable)	Double (F64)
dmv	Rychlostní výstup regulátoru (difference)	Double (F64)
de	Regulační odchylka	Double (F64)
SAT	Saturace	Bool
	off ... lineární zákon řízení	
	on výstup regulátoru je saturován	

POUT – Pulzní výstup

Symbol bloku

Licence: [STANDARD](#)



Popis funkce

Blok **POUT** tvaruje vstupní pulzy **U** takovým způsobem, že délka výstupního pulzu **Y** je alespoň **dtime** sekund a prodleva mezi dvěma sousedními výstupními pulzy je minimálně **btime** sekund. Vstupní pulz, který přijde po sestupné hraně výstupního signálu dříve, než uplyne čas **btime**, nezpůsobí žádnou odezvu na výstupu **Y**.

Tento blok propaguje kvalitu signálu. Více informací je uvedeno v sekci [1.4](#).

Vstup

U	Logický vstupní signál	Bool

Parametr

dtime	Minimální trvání výstupního pulzu [s]	⊙1.0	Double (F64)
btime	Minimální prodleva mezi pulzy [s]	⊙1.0	Double (F64)

Výstup

Y	Logický výstupní signál	Bool

PRGM – Programátor

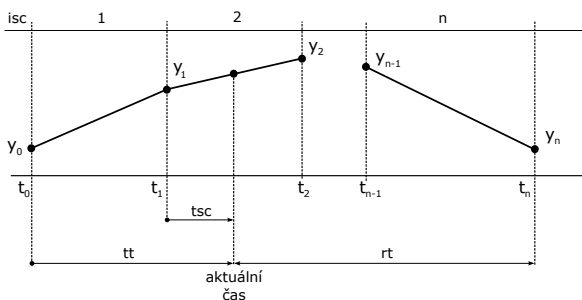
Symbol bloku

Licence: [STANDARD](#)



Popis funkce

Blok PRGM je určen pro generování časových funkcí (programů) složených z n lineárních částí definovaných $(n + 1)$ rozměrnými vektory $\mathbf{tm} = [t_0, \dots, t_n]$ času a požadovaných hodnot $\mathbf{y} = [y_0, \dots, y_n]$ (generovaná křivka je spojitá po částech lineární, viz. obrázek). Nejčastěji je používán pro generování požadované hodnoty regulátoru. Generování programu je spuštěno vstupem **RUN = on**; přechod zpět na **RUN = off** vrací stav programátoru do základního stavu. Vstup **DEF** nastaví **sp** na hodnotu **spv** a po vymizení hodnoty **DEF = on** se pokračuje přejetím po rampě na nejbližší následující uzel, čas přitom není narušen. Vstup **HLD = on** zmrazí výstupní hodnotu **sp** a všechny výstupní časy (**tsc**, **tt**, **rt**), po vymizení hodnoty **HLD = on** se pokračuje z okamžiku zmrazení dále podle programu. Je-li při přechodu **HLD on**→**off** nastaven vstup **CON = on**, nepokračuje se od okamžiku zmrazení, ale najede se do uzlového bodu s indexem **ind** po rampě za čas **trt**. Index uzlového bodu **ind** musí být rovný nebo větší než aktuálně prováděný sektor (v okamžiku **HLD on**→**off**). Je-li **RPT = on**, potom se program generuje opakovaně.



Tento blok nepropaguje kvalitu signálu. Více informací je uvedeno v sekci 1.4.

Vstup

RUN	Povolení běhu algoritmu	Bool
DEF	Inicializace sp na hodnotu spv	Bool
spv	Inicializační hodnota	Double (F64)
HLD	Zmrazení výstupu a výstupních časů	Bool

CON	Pokračování od uzlového bodu ind	Bool
ind	Index uzlového bodu pro pokračování	Long (I32)
trt	Čas pro dosažení požadovaného uzlu	Double (F64)
RPT	Opakování sekvence	Bool

Parametr

nmax	Rezervovaná paměť pro pole	↓4 ↑100000000 ⊙10	Long (I32)
tmunits	Jednotky pro zadávání časů	⊙1	Long (I32)
	1 sekundy		
	2 minuty		
	3 hodiny		
tm	Vektor vzestupně uspořádaných časů	⊙[0 1 2]	Double (F64)
y	Vektor hodnot časové funkce	⊙[0 1 0]	Double (F64)

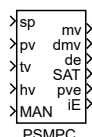
Výstup

sp	Požadovaná hodnota (setpoint)	Double (F64)
isc	Aktuální sektor funkce	Long (I32)
tsc	Čas od začátku sektoru	Double (F64)
tt	Čas od startu generování časové funkce	Double (F64)
rt	Čas do konce programu	Double (F64)
CNF	Příznak sledování nakonfigurované křivky	Bool
E	Chyba, časy uzlů nejsou seřazeny vzestupně	Bool

PSMPC – Prediktivní regulátor typu pulse-step

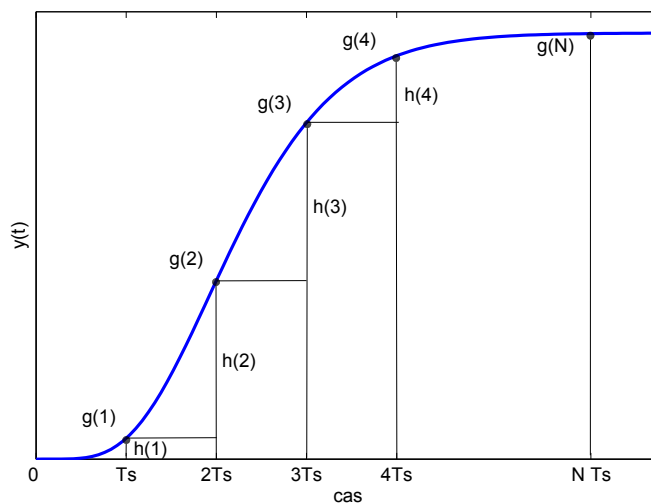
Symbol bloku

Licence: [ADVANCED](#)



Popis funkce

Funkční blok **PSMPC** (Pulse Step Model Predictive Control) je určen pro realizaci vysoce kvalitních regulátorů pro obtížně regulovatelné lineární časově invariantní soustavy s omezením akční veličiny (např. soustavy s dopravním zpožděním nebo s neminimální fází). Zvláště výhodný je pro případy, kdy je požadován velmi rychlý přechod z jedné hodnoty regulované veličiny na druhou bez překmitu. Regulátor **PSMPC** však může být obecně použit všude tam, kde je běžně nasazován standardní PID regulátor a kde žádáme vysokou kvalitu regulace.



PSMPC je prediktivní regulátor s explicitně zadaným intervalovým omezením akční veličiny. Pro účely predikce je použit model ve tvaru diskrétní přechodové charakteristiky $g(j)$, $j = 1, \dots, N$. Na obrázku výše je naznačen způsob, jakým lze tuto posloupnost získat ze spojitě přechodové charakteristiky. Poznamenejme, že N musí být zvoleno dostatečně velké, aby přechodová charakteristika byla popsána až do ustáleného stavu ($N \cdot T_S > t_{95}$, kde T_S je perioda vzorkování regulátoru a t_{95} je doba ustálení na 95 % konečné hodnoty).

Pro stabilní, lineární a t-invariantní systémy s monotónní přechodovou charakteristikou je alternativně možné použít momentový množinový model [7] a popsat systém pouze třemi charakteristickými čísly κ, μ a σ^2 , které je možno určit z jednoduchého pulzního experimentu. Řízený systém pak aproximujeme buď přenosem prvního řádu s dopravním zpožděním

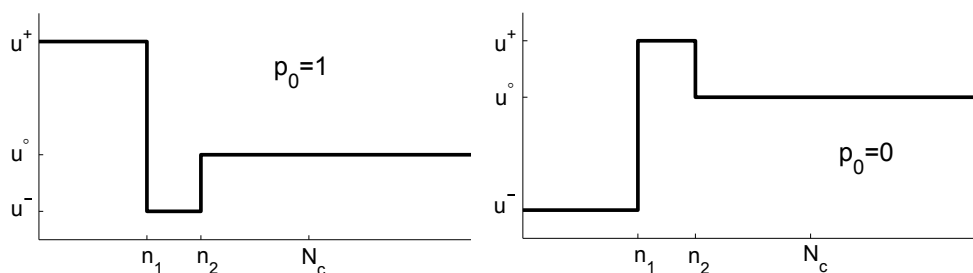
$$F_{FOPDT}(s) = \frac{K}{\tau s + 1} \cdot e^{-Ds}, \quad \kappa = K, \quad \mu = \tau + D, \quad \sigma^2 = \tau^2 \quad (7.1)$$

nebo přenosem druhého řádu s dopravním zpožděním

$$F_{SOPDT}(s) = \frac{K}{(\tau s + 1)^2} \cdot e^{-Ds}, \quad \kappa = K, \quad \mu = 2\tau + D, \quad \sigma^2 = 2\tau^2 \quad (7.2)$$

se stejnými charakteristickými čísly. Typ aproximace se zadává parametrem `imtype`.

Pro zjednodušení on-line optimalizace v otevřené smyčce je množina přípustných posloupností řízení omezena pouze na posloupnosti ve tvaru "pulz-skok" zobrazené na obrázku níže.



Poznamenejme, že každá taková posloupnost je jednoznačně určena jen třemi čísly $n_1, n_2 \in \{0, \dots, N_C\}$ a $u^\infty \in \langle u^-, u^+ \rangle$, kde $N_C \in \{0, 1, \dots\}$ je horizont řízení a u^-, u^+ označují po řadě zadanou dolní a horní mez akční veličiny regulátoru. On-line optimalizace (vzhledem k n_1, n_2 a u^∞) spočívá v minimalizaci kritéria

$$I = \sum_{i=N_1}^{N_2} \hat{e}(k+i|k)^2 + \lambda \sum_{i=0}^{N_C} \Delta \hat{u}(k+i|k)^2 \rightarrow \min, \quad (7.3)$$

kde $\hat{e}(k+i|k)$ je v kroku k predikovaná regulační odchylka na intervalu predikce $i \in \{N_1, N_2\}$, $\Delta \hat{u}(k+i|k)$ jsou difference řídicího signálu na intervalu $i \in \{0, N_C\}$ a λ je koeficient penalizace změn akční veličiny. Pro nalezení optima úlohy (7.3) je použita kombinace metody nejmenších čtverců a hrubé síly. Hodnota u^∞ je určena pro všechny přípustné kombinace p_0, n_1 a n_2 a následně je z nich vybrána optimální řídicí sekvence pro řízení v otevřené smyčce. Ve skutečnosti je však vždy aplikován pouze první krok této řídicí sekvence a v další vzorkovací periodě je celý optimalizační proces zopakován. Tím se řídicí strategie mění na zpětnovazební řízení.

Parametry prediktivního regulátoru, kromě modelu řízené soustavy a omezení jejího vstupu, jsou horizont řízení N_C , horizont predikce N_1, N_2 a koeficient λ . Pouze poslední uvedený parametr λ je určen pro ruční doladění kvality regulace při rutinním uvádění

do provozu. V případě použití modelu soustavy ve tvaru přenosu (7.1) nebo (7.2) jsou parametry N_1 , N_2 zvoleny automaticky na základě charakteristických čísel μ, σ^2 . Regulator potom může být efektivně laděn „ručně“ pouze seřizováním charakteristických čísel procesu κ, μ, σ^2 .

Varování

Při použití bloku **PSMPC** pro simulaci v systému Matlab/Simulink je třeba zajistit, aby pole **sr** bylo dostatečně velké, tak aby jím definovaný buffer pojmul interně vygenerovanou přechodovou charakteristiku určenou z FOPDT nebo SOPDT modelu. V opačném případě blok hlásí chybu.

Tento blok propaguje kvalitu signálu. Více informací je uvedeno v sekci 1.4.

Vstup

sp	Požadovaná hodnota (setpoint)	Double (F64)
pv	Řízená veličina	Double (F64)
tv	Veličina pro vysledování	Double (F64)
hv	Hodnota výstupu v manuálním režimu	Double (F64)
MAN	Manuální nebo automatický režim	Bool
	off ... automatický režim	
	on manuální režim	

Parametr

nc	Délka horizontu řízení	⊙5	Long (I32)
np1	Začátek koincidenčního intervalu	⊙1	Long (I32)
np2	Konec koincidenčního intervalu	⊙10	Long (I32)
lambda	Koeficient penalizace změn řízení	⊙0.05	Double (F64)
umax	Horní mez akčního zásahu regulátoru	⊙1.0	Double (F64)
umin	Dolní mez akčního zásahu regulátoru	⊙-1.0	Double (F64)
imtype	Typ modelu řízené soustavy	⊙3	Long (I32)
	1 model prvního řádu		
	2 model druhého řádu		
	3 přechodová charakteristika		
kappa	Statické zesílení	⊙1.0	Double (F64)
mu	Míra zpoždění soustavy	⊙20.0	Double (F64)
sigma	Míra délky odezvy soustavy	⊙10.0	Double (F64)
nmax	Rezervovaná paměť pro pole	↓10 ↑10000 ⊙32	Long (I32)
sr	Diskrétní přechodová charakteristika		Double (F64)
	⊙[0 0.2642 0.5940 0.8009 0.9084 0.9596 0.9826 0.9927 0.9970 0.9988 0.9995]		

Výstup

mv	Akční zásah regulátoru (manipulated variable)	Double (F64)
dmv	Rychlostní výstup regulátoru (difference)	Double (F64)

de	Regulační odchylka	Double (F64)
SAT	Saturace	Bool
	off ... lineární zákon řízení	
	on výstup regulátoru je satureván	
pve	Odhad řízené veličiny	Double (F64)
iE	Kód chyby	Long (I32)
	0 bez chyby	
	1 nesprávný FOPDT model	
	2 nesprávný SOPDT model	
	3 chyba v zadání přechodové charakteristiky	

PWM – Pulzně-šířková modulace

Symbol bloku

Licence: **STANDARD**



Popis funkce

Blok PWM provádí pulzně šířkovou modulaci vstupního signálu z intervalu od -1 do $+1$. Užitím tohoto bloku je možné realizovat proporcionální akční veličinu i u akčních členů s jedním (např. topení zapnuto/vypnuto) nebo dvěma (např. topení zapnuto/vypnuto a chlazení zap./vyp.) binárními vstupy. Šířka L výstupního pulzu je určena vztahem:

$$L = \text{pertm} * |u|,$$

kde pertm je perioda modulace. Je-li $u > 0$ ($u < 0$), pulz je generován na výstupu UP (DN). Z praktických důvodů je však délka generovaného pulzu dále upravována podle zadaných parametrů bloku. Faktor asymetrie asyfac definuje poměr mezi délkou negativního pulzu DN a délkou pozitivního pulzu UP. Modifikované délky se počítají podle vztahů:

$$\begin{aligned} \text{jestliže } u > 0 \text{ potom } L(\text{UP}) &:= \begin{cases} L & \text{pro } \text{asyfac} \leq 1.0 \\ L/\text{asyfac} & \text{pro } \text{asyfac} > 1.0 \end{cases} \\ \text{jestliže } u < 0 \text{ potom } L(\text{DN}) &:= \begin{cases} L * \text{asyfac} & \text{pro } \text{asyfac} \leq 1.0 \\ L & \text{pro } \text{asyfac} > 1.0 \end{cases} \end{aligned}$$

kteří pro libovolnou hodnotu $\text{asyfac} > 0$ zajišťují, že maximální délka generovaných pulzů je rovna pertm . Dále, jestliže vypočtená délka pulzu je menší než dtime , potom je výsledná délka nastavena na nulu. Jestliže se vypočtená délka pulzu liší od pertm méně než btime , potom je výsledná délka nastavena na pertm . Jestliže kladný pulz UP je následovaný záporným pulzem DN nebo obráceně, potom pozdější pulz je v případě potřeby posunut tak, že vzdálenost mezi těmito dvěma pulzy je alespoň offtime . Jestliže $\text{SYNCH} = \text{on}$, potom změna vstupu u způsobí okamžitý přepočtení délky výstupního pulzu za předpokladu, že není splněna synchronizační podmínka mezi začátkem periody modulace a okamžikem změny vstupu u .

Tento blok propaguje kvalitu signálu. Více informací je uvedeno v sekci 1.4.

Vstup

u

Analogový vstupní signál

Double (F64)

Parametr

<code>pertm</code>	Perioda šířkové modulace [s]	⊙10.0	Double (F64)
<code>dtime</code>	Minimální trvání výstupního pulzu [s]	⊙0.1	Double (F64)
<code>btime</code>	Minimální prodleva mezi pulzy [s]	⊙0.1	Double (F64)
<code>offtime</code>	Minimální prodleva mezi pulzy opačné polarity [s]	⊙1.0	Double (F64)
<code>asyfac</code>	Faktor asymetrie	⊙1.0	Double (F64)
<code>SYNCH</code>	Synchronizační příznak pro začátek periody		Bool
	<code>off</code> ... synchronizace vypnuta		
	<code>on</code> synchronizace zapnuta		

Výstup

<code>UP</code>	Signál UP (nahoru, více)	Bool
<code>DN</code>	Signál DN (dolů, méně)	Bool

RLY – Relé s hysterezí

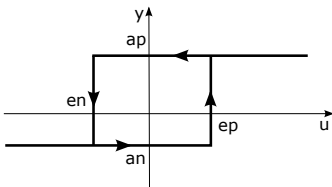
Symbol bloku

Licence: [STANDARD](#)



Popis funkce

Blok RLY transformuje vstupní analogový signál u na výstupní analogový signál y podle níže uvedeného obrázku.



Tento blok propaguje kvalitu signálu. Více informací je uvedeno v sekci [1.4](#).

Vstup

u	Analogový vstupní signál	Double (F64)

Parametr

ep	Mez pro zapnutí	$\odot 1.0$	Double (F64)
en	Mez pro vypnutí	$\odot -1.0$	Double (F64)
ap	Výstup ve stavu ZAPNUTO	$\odot 1.0$	Double (F64)
an	Výstup ve stavu VYPNUTO	$\odot -1.0$	Double (F64)
$y0$	Počáteční hodnota výstupu		Double (F64)

Výstup

y	Analogový výstupní signál	Double (F64)

SAT – Saturace výstupu s proměnnými mezemi

Symbol bloku

Licence: [STANDARD](#)



Popis funkce

Blok SAT kopíruje vstup u na výstupu y , pokud pro vstupní veličinu platí $\text{lolim} \leq u$ a $u \leq \text{hilim}$, kde lolim a hilim jsou stavové proměnné bloku. Je-li $u < \text{lolim}$ (resp. $u > \text{hilim}$), potom $y = \text{lolim}$ ($y = \text{hilim}$). Horní a dolní limit jsou buď pevné hodnoty dané po řadě parametry bloku hilim0 a lolim0 (případ $\text{HLD} = \text{on}$) nebo jsou řízeny vstupy hi a lo ($\text{HLD} = \text{off}$). Maximální rychlost změny aktivních mezí hilim a lolim je dána časovými konstantami tp a tn . Parametr tp určuje maximální kladnou strmost a tn maximální zápornou strmost změn hilim a lolim . Omezení strmosti změn mezí je aktivní i v případě, že hodnoty mezí měníme ručně ($\text{HLD} = \text{on}$) pomocí parametrů hilim0 a lolim0 . Pro možnost okamžitých změn saturačních mezí je potřeba nastavit $tp = 0$ a $tn = 0$. Výstupy HL a LL signalizují po řadě horní a dolní saturaci.

Pokud je to potřeba, parametry hilim0 a lolim0 jsou použity jako počáteční hodnoty pro saturační meze řízené vstupními signály.

Tento blok propaguje kvalitu signálu. Více informací je uvedeno v sekci [1.4](#).

Vstup

u	Analogový vstupní signál	Double (F64)
hi	Horní saturační mez	Double (F64)
lo	Dolní saturační mez	Double (F64)

Parametr

tp	Omezení rychlosti změny aktivních mezí v kladném směru	Double (F64)
	⊙1.0	
tn	Omezení rychlosti změny aktivních mezí v záporném směru	Double (F64)
	⊙1.0	
hilim0	Horní omezení výstupu (platné pro $\text{HLD}=1$)	⊙1.0 Double (F64)
lolim0	Dolní omezení výstupu (platné pro $\text{HLD}=1$)	⊙-1.0 Double (F64)
HLD	Pevné saturační meze	⊙on Bool
	off ... proměnné meze	
	on pevné meze	

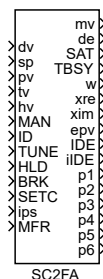
Výstup

y	Analogový výstupní signál	Double (F64)
HL	Příznak saturace na horní mezi	Bool
LL	Příznak saturace na dolní mezi	Bool

SC2FA – Stavový regulátor systému 2. řádu s autotunerem

Symbol bloku

Licence: [AUTOTUNING](#)



Popis funkce

Funkční blok **SC2FA** realizuje stavový regulátor pro systém druhého řádu (7.4) s frekvenčním autotunerem. Je vhodný především pro aktivní řízení (zatlumení) kmitavých systémů s velmi slabým tlumením ($\xi < 0,1$). Může však být použit též jako samonastavující se regulátor pro libovolný systém, který lze s dostatečnou přesností popsat přenosem ve tvaru

$$F(s) = \frac{b_1 s + b_0}{s^2 + 2\xi\Omega s + \Omega^2}, \quad (7.4)$$

kde $\Omega > 0$ je přirozená (netlumená) frekvence, ξ , $0 < \xi < 1$, je koeficient tlumení a b_1 , b_0 jsou libovolná reálná čísla. Blok pracuje ve dvou režimech, v režimu „Identifikace a návrh“ a v režimu „Regulace“.

Identifikace a návrh

Režim „Identifikace a návrh“ se zapíná nastavením binárního vstupu **ID** = **on**. Během identifikačního experimentu jsou naměřeny dva body frekvenční charakteristiky s daným fázovým zpožděním. Vlastní proces identifikace a návrhu se spouští náběžnou hranou **off**→**on** vstupu **TUNE**. Na výstupu bloku **mv** se poté objeví budící harmonický signál se stejnou složkou **ubias**, amplitudou **uamp** a frekvencí ω postupně probíhající interval $\langle \mathbf{wb}, \mathbf{wf} \rangle$. Aktuální frekvence ω je přitom kopírována na výstup **w**. Rychlost změny (rozmítání) frekvence je dána parametrem **cp**, který udává relativní zmenšení počáteční periody $T_b = \frac{2\pi}{\mathbf{wb}}$ budící sinusovky za čas T_b , tedy

$$c_p = \frac{\mathbf{wb}}{\omega(T_b)} = \frac{\mathbf{wb}}{\mathbf{wbe}^{\gamma T_b}} = e^{-\gamma T_b}. \quad (7.5)$$

Hodnota parametru **cp** se obvykle pohybuje v intervalu $\mathbf{cp} \in \langle 0,95; 1 \rangle$. Čím menší je koeficient tlumení řízeného systému, tím více se musí **cp** blížit k jedné.

Na začátku identifikace má budící signál frekvenci $\omega = \mathbf{wb}$. Po uplynutí **stime** startuje výpočet odhadu aktuálního bodu frekvenční charakteristiky. Jeho reálná a imaginární

část se průběžně kopíruje po řadě na výstupy `xre` a `xim`. Je-li parametr bloku `MANF` nastaven na 0, potom se v procesu identifikace dvakrát zastaví rozmítání frekvence na dobu `stime` a to v okamžicích, kdy jsou poprvé dosaženy body s fázovým zpožděním `ph1` a `ph2`. Přednastavené hodnoty parametrů `ph1` a `ph2` jsou po řadě -60° a -120° a mohou být změněny na libovolné hodnoty v intervalu $(-360^\circ, 0^\circ)$, přičemž `ph1` > `ph2`. Po uplynutí `stime` sekund při zastavení ve fázi `ph1`, resp. `ph2` se spočítá průměr posledních `iavg` naměřených bodů (průměrováním tedy získáme odhad příslušného bodu frekvenční charakteristiky) pro následný výpočet parametrického modelu ve tvaru (7.4). Je-li `MANF = on`, potom je možné provést „navzorkování“ dvou bodů frekvenční charakteristiky ručně pomocí vstupu `HLD`. Vstup `HLD = on` zastaví rozmítání frekvence a opětovné nastavení `HLD = off` vede k jeho pokračování. Ostatní funkce jsou identické.

V případě potřeby je možné proces identifikace přerušit vstupem `BRK = on`. Jsou-li již v tomto okamžiku oba dva body pro parametrickou identifikaci určeny, pokračuje se v návrhu regulátoru normálním způsobem. V opačném případě je proces ukončen bez návrhu regulátoru a výstup `IDE = on` signalizuje chybu.

Během vlastní „identifikace a návrhu“:

- je výstup `TBSY` nastaven na 1. Po skončení je shozen na 0.
- Při bezchybném návrhu regulátoru je výstup `IDE = off` a výstup `iIDE` signalizuje jednotlivé fáze identifikačního experimentu:
 - Přibližování k prvnímu bodu je `iIDE = -1`,
 - zastavení v prvním bodě `iIDE = 1`,
 - přibližování k druhému bodu je `iIDE = -2`,
 - zastavení v druhém bodě `iIDE = 2` a
 - poslední fáze po zastavení v druhém bodě je `iIDE = -3`.
- Jestliže identifikace skončí s chybou, pak je `IDE = on` a číslo na výstupu `iIDE` specifikuje příslušnou chybu. Viz popis parametru `iIDE` níže.

Vypočtené parametry stavového regulátoru jsou instalovány okamžitě do algoritmu řízení, jestliže vstup `SETC` je trvale nastaven na `on`. V opačném případě se provede nastavení parametrů až po ukončení návrhu na náběžnou hranu vstupu `SETC`. Výsledky parametrické identifikace a návrhu stavového regulátoru je možné získat na výstupech `p1`, `p2`, ..., `p6` vhodným nastavením vstupu `ips`. Pro jednotlivé hodnoty `ips` mají parametry tyto významy:

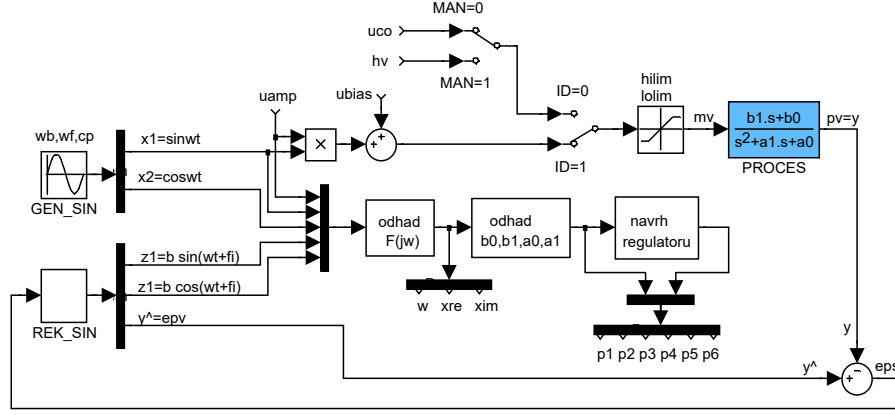
- **0:** Dva body frekvenční charakteristiky
 - `p1` ... frekvence 1. měřeného bodu v rad/s
 - `p2` ... reálná část 1. bodu
 - `p3` ... imaginární část 1. bodu
 - `p4` ... frekvence 2. měřeného bodu v rad/s

- p5 ... reálná část 2. bodu
- p6 ... imaginární část 2. bodu
- **1:** Model druhého řádu ve formě (7.6)
 - p1 ... parametr b_1
 - p2 ... parametr b_0
 - p3 ... parametr a_1
 - p4 ... parametr a_0
- **2:** Model druhého řádu ve formě (7.7)
 - p1 ... parametr K_0
 - p2 ... parametr τ
 - p3 ... parametr Ω v rad/s
 - p4 ... parametr ξ
 - p5 ... parametr Ω v Hz
 - p6 ... rezonanční frekvence v Hz
- **3:** Parametry stavové zpětné vazby
 - p1 ... parametr f_1
 - p2 ... parametr f_2
 - p3 ... parametr f_3
 - p4 ... parametr f_4
 - p5 ... parametr f_5

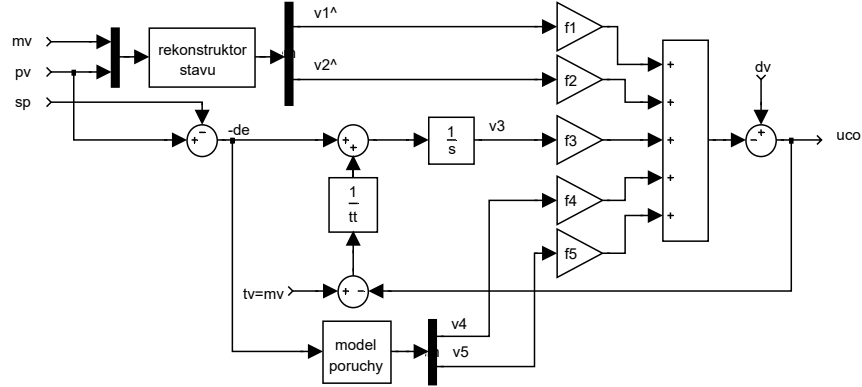
Náběžná hrana na vstupu **MFR** po skončení identifikace (**TBSY** = **off**) odstartuje generování frekvenční charakteristiky získaného parametrického modelu na výstupech **w**, **xre**, **xim**. Takto je možno porovnat její průběh s „přímo odměřenou“ frekvenční charakteristikou systému.

Regulace

V režimu „regulace“ (binární vstup **ID** = **off**) může regulátor pracovat v manuálním módu (**MAN** = **on**) nebo v automatickém módu. Jestliže je blok regulátoru spuštěn (při studeném startu) s hodnotou vstupu **ID** = **off**, potom se předpokládá, že zadané parametry bloku **mb0**, **mb1**, **ma0** a **ma1** odpovídají dříve určeným koeficientům b_0 , b_1 , a_0 a a_1 přenosu řízeného systému a automaticky proběhne návrh stavového regulátoru. Je-li regulátor navíc v automatickém módu a **SETC** = **on**, potom zákon řízení od počátku využívá nově navržené parametry. Takto lze vypustit identifikaci při opakovaném spuštění bloku.



Na výše uvedeném obrázku je zjednodušené vnitřní schéma samonastavujícího se stavového regulátoru, část frekvenční identifikace. Na spodním obrázku je stavová zpětná vazba s rekonstruktorem stavu a ošetřením unášení integrační složky. Na obrázku není znázorněna skutečnost, že blok návrhu regulátoru v části frekvenční identifikace automaticky nastaví parametry rekonstruktoru stavu a koeficienty $f_1, f_2, \dots, f_5$ stavové zpětné vazby.



Model řízeného systému je brán jako systém 2. řádu s přenosem ve tvaru (7.4). Jednoduchými úpravami lze dojít k přenosům

$$F(s) = \frac{(b_1 s + b_0)}{s^2 + a_1 s + a_0} \quad (7.6)$$

a

$$F(s) = \frac{K_0 \Omega^2 (\tau s + 1)}{s^2 + 2\xi \Omega s + \Omega^2}. \quad (7.7)$$

Parametry těchto přenosů je možné po skončení identifikace přechíst z výstupů $p_1, \dots, p_6$. Význam těchto výstupů se mění při změně vstupu ips , avšak pouze pokud neběží identifikace (tedy $TBSY = off$).

Tento blok propaguje kvalitu signálu. Více informací je uvedeno v sekci 1.4.

Vstup

dv	Proměnná dopředné vazby	Double (F64)
sp	Požadovaná hodnota (setpoint)	Double (F64)
pv	Řízená veličina	Double (F64)
tv	Veličina pro vysledování	Double (F64)
hv	Hodnota výstupu v manuálním režimu	Double (F64)
MAN	Manuální nebo automatický režim	Bool
	off ... automatický režim	
	on manuální režim	
ID	Režim identifikace nebo regulace	Bool
	off ... režim regulátoru	
	on režim identifikace a návrhu	
TUNE	Zahájení ladicího experimentu	Bool
HLD	Zastavení rozmítání frekvence	Bool
BRK	Signál pro přerušení	Bool
SETC	Přijmutí a nastavení parametrů regulátoru	Bool
	off ... parametry jsou pouze vypočítány	
	on parametry jsou přijaty ihned po vypočtení	
	off->on jednorázové přijetí vypočtených parametrů	
ips	Význam výstupních signálů	Long (I32)
	0 dva body frekvenční charakteristiky	
	1 model druhého řádu (obecný)	
	2 model druhého řádu (frekvence)	
	3 parametry stavové zpětné vazby	
MFR	Generování frekvenční charakteristiky modelu	Bool

Parametr

ubias	Stejnoseměrná složka budicího signálu	Double (F64)
uamp	Amplituda budicího signálu	⊙1.0 Double (F64)
wb	Počátek frekvenčního intervalu [rad/s]	⊙1.0 Double (F64)
wf	Konec frekvenčního intervalu [rad/s]	⊙10.0 Double (F64)
isweep	Způsob rozmítání frekvence	⊙1 Long (I32)
	1 logaritmické	
	2 lineární	
	3 kombinované	
cp	Rychlost rozmítání	↓0.5 ↑1.0 ⊙0.995 Double (F64)
iavg	Počet vzorků pro průměrování	⊙10 Long (I32)
alpha	Relativní poloha pólů rekonstruktoru (ident.)	⊙2.0 Double (F64)
xi	Koeficient tlumení rekonstruktoru (ident.)	⊙0.707 Double (F64)
MANF	Ruční výběr bodů frekvenční charakteristiky	Bool
	off ... zakázáno	
	on povoleno	
ph1	Fázové zpoždění prvního bodu (ve stupních)	⊙-60.0 Double (F64)
ph2	Fázové zpoždění druhého bodu (ve stupních)	⊙-120.0 Double (F64)

stime	Doba ustálení [s]	⊙10.0	Double (F64)
ralpha	Relativní poloha pólů rekonstruktoru	⊙4.0	Double (F64)
rxl	Koeficient tlumení rekonstruktoru	⊙0.707	Double (F64)
acl1	Relativní poloha 1. dvojice pólů uzavř. smyčky	⊙1.0	Double (F64)
xicl1	Tlumení 1. dvojice pólů uzavřené smyčky	⊙0.707	Double (F64)
INTGF	Příznak rozšíření o integrátor	⊙on	Bool
	off ... stavový model neobsahuje integrátor		
	on integrátor je zahrnut ve stavovém modelu		
apcl	Relativní poloha reálného pólu	⊙1.0	Double (F64)
DISF	Příznak rozšíření o model poruchy		Bool
	off ... stavový model neobsahuje model poruchy		
	on model poruchy je zahrnut ve stavovém modelu		
dom	Přirozená frekvence modelu poruchy [rad/s]	⊙1.0	Double (F64)
dxi	Koeficient tlumení modelu poruchy		Double (F64)
acl2	Relativní poloha 2. dvojice	⊙2.0	Double (F64)
xicl2	Tlumení 2. dvojice pólů uzavřené smyčky	⊙0.707	Double (F64)
tt	Časová konstanta výsledování	⊙1.0	Double (F64)
hilim	Horní mez akčního zásahu regulátoru	⊙1.0	Double (F64)
lolim	Dolní mez akčního zásahu regulátoru	⊙-1.0	Double (F64)
mb1p	Koeficient přenosu řízeného systému (b1)		Double (F64)
mb0p	Koeficient přenosu řízeného systému (b0)	⊙1.0	Double (F64)
ma1p	Koeficient přenosu řízeného systému (a1)	⊙0.2	Double (F64)
ma0p	Koeficient přenosu řízeného systému (a0)	⊙1.0	Double (F64)

Výstup

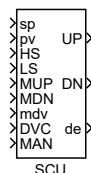
mv	Akční zásah regulátoru (manipulated variable)	Double (F64)
de	Regulační odchylka	Double (F64)
SAT	Saturace	Bool
	off ... lineární zákon řízení	
	on výstup regulátoru je saturován	
TBSY	Příznak probíhajícího ladicího experimentu	Bool
	off ... identifikace neprobíhá	
	on běží identifikační experiment	
w	Odhad bodu frekvenční charakteristiky - frekvence [rad/s]	Double (F64)
xre	Odhad bodu frekvenční charakteristiky - reálná část	Double (F64)
xim	Odhad bodu frekvenční charakteristiky - imaginární část	Double (F64)
epv	Rekonstruovaný signál pv	Double (F64)
IDE	Příznak chyby identifikace	Bool
	off ... identifikace proběhla v pořádku	
	on identifikace skončila s chybou	

iIDE	Kód chyby	Long (I32)
	101 ... vzorkovací frekvence je příliš nízká	
	102 ... chyba identifikace bodu frekvenční charakteristiky	
	103 ... saturace výstupu během identifikace	
	104 ... nesprávný model procesu	
p1..p6	Výsledky identifikace a návrhu regulátoru	Double (F64)

SCU – Krokový regulátor s polohovou zpětnou vazbou

Symbol bloku

Licence: [STANDARD](#)



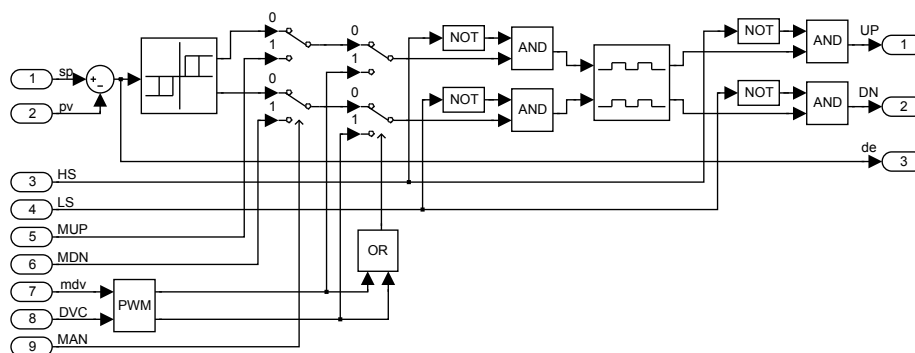
Popis funkce

Blok SCU je polohový regulátor servoventilu s třístavovým výstupem. Ve spojení s nadřazeným blokem [PIDU](#) nebo od něho odvozeným ([PIDMA](#), atd.) je určen k realizaci třístavového krokového regulátoru s polohovou zpětnou vazbou.

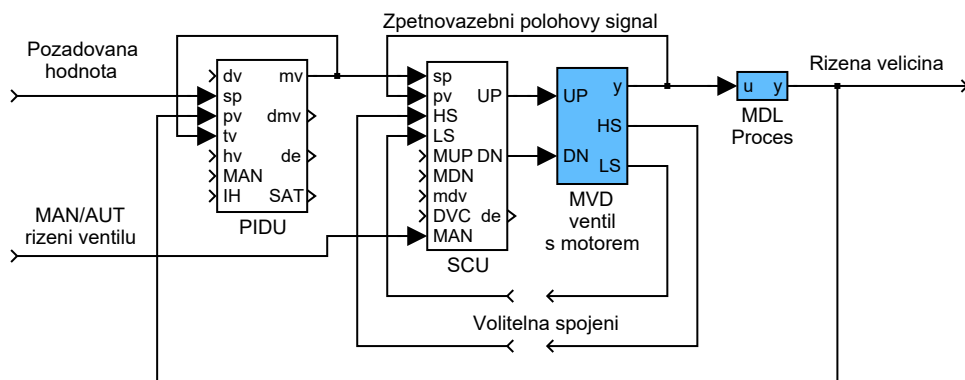
Blok SCU nejprve zpracovává regulační odchylku $sp - pv$ na třístavový výstup symetrickým třístavovým algoritmem s parametry (prahy) $thron$ a $throff$ (viz blok [TSE](#), uvažujte parametry $ep = thron$, $epoff = throff$, $en = -thron$ a $enoff = -throff$). Parametr $trun$ určuje dobu, po kterou se poloha motoru změní o jednu jednotku. Parametr $RACT$ určuje, pro kterou polaritu odchylky jsou generovány pulsy UP (více) nebo DN (méně). Výstupy symetrického třístavového algoritmu jsou dále zpracovávány tak, že délka libovolného generovaného pulsu (UP, DN) na výstupu bloku je alespoň $dtime$ a prodleva mezi dvěma po sobě jdoucími pulsy opačné polarity je alespoň $btime$. Jsou-li dostupné signály od koncových spínačů servoventilu, potom by měly být připojeny na vstupy HS (horní spínač) a LS (dolní spínač).

K dispozici je také sada vstupů pro manuální ovládání. Přepnutí do manuálního režimu je možné pomocí vstupu $MAN = on$, pak lze s motorem pohybovat tam a zpět pomocí signálů MUP a MDN, eventuálně lze pomocí vstupu mdv nastavit, o kolik se má změnit poloha motoru, a tento požadavek potvrdit náběžnou hranou ($off \rightarrow on$) na vstupu DVC.

Celková funkce bloku SCU je dostatečně zřejmá z následujícího diagramu.



Úplný třístavový krokový regulátor s polohovou zpětnou vazbou je zobrazen na následujícím obrázku.



Tento blok propaguje kvalitu signálu. Více informací je uvedeno v sekci 1.4.

Vstup

sp	Požadovaná hodnota (výstup primárního regulátoru)	Double (F64)
pv	Rízená veličina (poloha servopohonu ventilu)	Double (F64)
HS	Horní koncový spínač	Bool
LS	Dolní koncový spínač	Bool
MUP	Manuální povel UP (přidej)	Bool
MDN	Manuální povel DN (uber)	Bool
mdv	Ruční diferenční hodnota	Double (F64)
DVC	Přijetí ruční diferenční hodnoty	Bool
MAN	Manuální nebo automatický režim	Bool
	off ... automatický režim	
	on manuální režim	

Parametr

thron	Mez pro zapnutí	↓0.0 ⊕0.02	Double (F64)
throff	Mez pro vypnutí	↓0.0 ⊕0.01	Double (F64)
dtime	Minimální trvání výstupního pulzu [s]	↓0.0 ⊕0.1	Double (F64)
btime	Minimální prodleva mezi pulzy [s]	↓0.0 ⊕0.1	Double (F64)
RACT	Převrácené působení výstupu regulátoru		Bool
	off ... vyšší mv -> vyšší pv		
	on vyšší mv -> nižší pv		
trun	Časová konstanta motoru	↓0.0 ⊕10.0	Double (F64)

Výstup

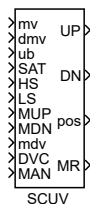
UP	Signál UP (nahoru, více)	Bool

DN	Signál DN (dolů, méně)	Bool
de	Regulační odchylka	Double (F64)

SCUV – Krokový regulátor s rychlostním výstupem

Symbol bloku

Licence: [STANDARD](#)



Popis funkce

Blok **SCUV** nahrazuje polohový regulátor **SCU** v úplné regulační smyčce s třístavovým krokovým regulátorem, jestliže polohový signál servoventilu není dostupný anebo dostatečně spolehlivý. Nadřazený regulátor **PIDU** (nebo odvozený) je propojen s blokem **SCUV** signály **mv**, **dmv** a **SAT**.

Jestliže je nadřazený regulátor typu **PI** nebo **PID** (**CWOI** = **off**), potom jsou všechny tři vstupy **mv**, **dmv** a **SAT** bloku **SCUV** zpracovávány speciálním integračním algoritmem a symetrickým třístavovým algoritmem s parametry (prahy) **thron** a **throff** (viz blok **TSE**, uvažujte parametry **ep** = **thron**, **epoff** = **throff**, **en** = **-thron** a **enoff** = **-throff**). Vzniklé pulsy (více, méně) jsou dále upravovány tak, že délka libovolného generovaného pulsu (**UP**, **DN**) na výstupu bloku je alespoň **dtime** a prodleva mezi dvěma po sobě jdoucími pulsy opačné polarity je alespoň **btime**. Parametr **RACT** určuje směr otáčení motoru. Poznamenejme, že nadřazený regulátor **PIDU** musí mít nastavení **icotype** = 4. Blok **SCUV** rekonstruuje rychlostní výstup nadřazeného regulátoru ze vstupů **mv** a **dmv**. Navíc, jestliže regulační odchylka nadřazeného regulátoru je menší než pásmo necitlivosti (**SAT** = **on**), potom je výstup vnitřního integrátoru bloku **SCUV** nulován. Takto je dosaženo klidu servoventilu při dostatečně malé regulační odchylce nadřazeného regulátoru ($|de| < dz$ – viz popis bloku **PIDU**).

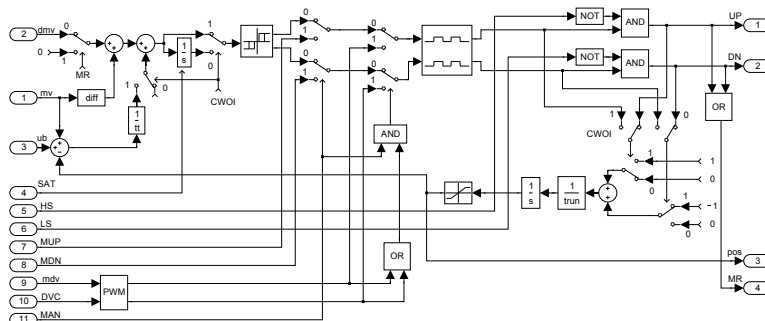
Poloha servoventilu **pos** je odhadována dalším vnitřním integrátorem s časovou konstantou **trun**. Jsou-li dostupné signály od koncových spínačů servoventilu, potom by měly být připojeny na vstupy **HS** (horní spínač) a **LS** (dolní spínač).

Jestliže je nadřazený regulátor typu **P** nebo **PD** (**CWOI** = **on**), potom může být regulační odchylka nadřazeného regulátoru manuálně odstraněna vhodným nastavením vstupu **ub**. V tomto případě je řídicí algoritmus bloku **SCUV** lehce modifikován. Je užita rekonstruovaná hodnota polohy servoventilu **pos** a parametry **thron**, **throff** a **tt** musí být pečlivě nastaveny pro potlačení střídání pulsů více a méně v ustáleném stavu.

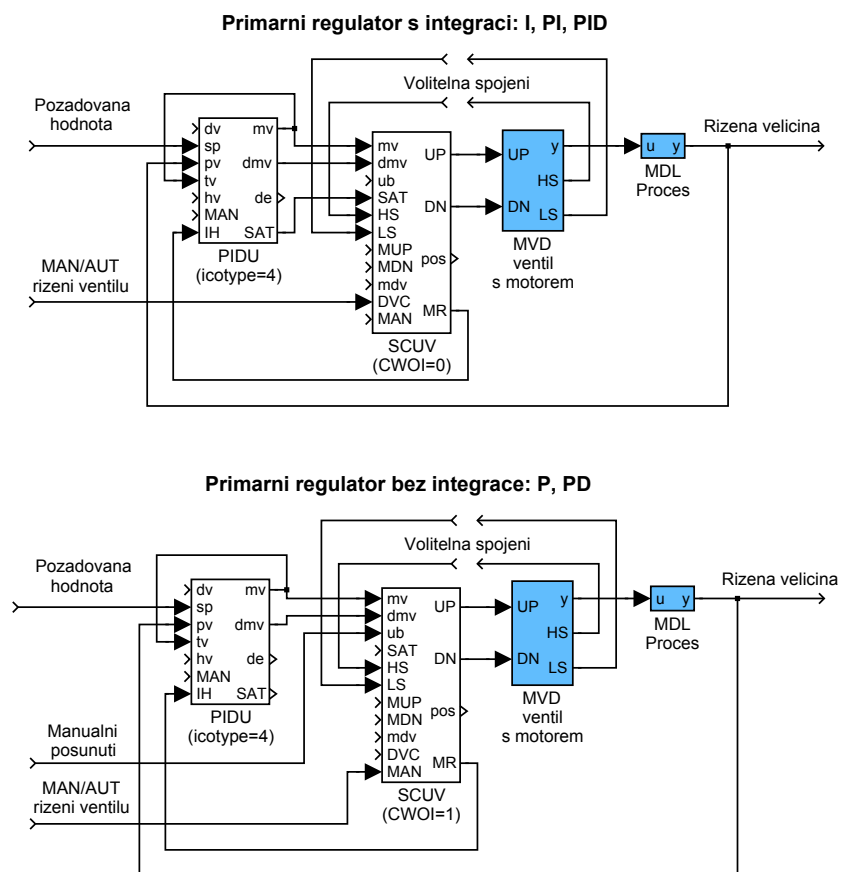
K dispozici je také sada vstupů pro manuální ovládání. Přepnutí do manuálního režimu je možné pomocí vstupu **MAN** = **on**, pak lze s motorem pohybovat tam a zpět pomocí signálů **MUP** a **MDN**, eventuálně lze pomocí vstupu **mdv** nastavit, o kolik se má změnit poloha motoru, a tento požadavek potvrdit náběžnou hranou (**off**→**on**) na vstupu

DVC.

Celková funkce bloku SCUV je zřejmá z následujícího diagramu:



Úplné třístavové krokové regulátory bez polohové zpětné vazby jsou zobrazeny na následujících obrázcích:



Tento blok propaguje kvalitu signálu. Více informací je uvedeno v sekci 1.4.

Vstup

mv

Akční zásah regulátoru (manipulated variable)

Double (F64)

dmv	Rychlostní výstup regulátoru (difference)	Double (F64)
ub	Posunutí (jen pokud je primární regulátor typu P nebo PD)	Double (F64)
SAT	Nulování interního integrátoru	Bool
HS	Horní koncový spínač	Bool
LS	Dolní koncový spínač	Bool
MUP	Manuální povel UP (přidej)	Bool
MDN	Manuální povel DN (uber)	Bool
mdv	Ruční diferenční hodnota	Double (F64)
DVC	Přijetí ruční diferenční hodnoty	Bool
MAN	Manuální nebo automatický režim	Bool
	off ... automatický režim	
	on manuální režim	

Parametr

thron	Mez pro zapnutí	↓0.0 ⊕0.02	Double (F64)
throff	Mez pro vypnutí	↓0.0 ⊕0.01	Double (F64)
dtime	Minimální trvání výstupního pulzu [s]	↓0.0 ⊕0.1	Double (F64)
btime	Minimální prodleva mezi pulzy [s]	↓0.0 ⊕0.1	Double (F64)
RACT	Převrácené působení výstupu regulátoru		Bool
	off ... vyšší mv -> vyšší pv		
	on vyšší mv -> nižší pv		
trun	Časová konstanta motoru	↓0.0 ⊕10.0	Double (F64)
CW0I	Regulátor bez integrační složky		Bool
	off ... regulátor s integrátorem (I, PI, PID)		
	on regulátor bez integrátoru (P, PD)		
tt	Časová konstanta vysledování	↓0.0 ⊕1.0	Double (F64)

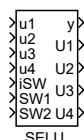
Výstup

UP	Signál UP (nahoru, více)	Bool
DN	Signál DN (dolů, méně)	Bool
pos	Simulovaná poloha motoru	Double (F64)
MR	Požadavek na běh motoru	Bool
	off ... motor neběží (UP=off a DN=on)	
	on motor se má pohybovat (UP=on nebo DN=on)	

SELU – Selektor aktivního regulátoru

Symbol bloku

Licence: [STANDARD](#)



Popis funkce

Blok **SELU** je určen pro přepínání aktivního regulátoru v případě selektorové regulace. Provádí výběr jednoho ze vstupních signálů **u1**, **u2**, **u3**, **u4** a kopíruje ho na výstup **y** buď podle celočíselného vstupu **iSW** (je-li parametr bloku **BINF** = **off**) nebo podle binárních vstupů **SW1** a **SW2** (**BINF** = **on**) dle následující tabulky.

iSW	SW1	SW2	y	U1	U2	U3	U4
0	off	off	u1	off	on	on	on
1	off	on	u2	on	off	on	on
2	on	off	u3	on	on	off	on
3	on	on	u4	on	on	on	off

Z této tabulky je patrný též význam logických výstupů **U1**, **U2**, **U3**, **U4**, které se používají jako vstupy bloků **SWU** pro realizaci funkce vysledování neaktivních regulátorů v selektorové regulaci.

Tento blok propaguje kvalitu signálu. Více informací je uvedeno v sekci [1.4](#).

Vstup

u1..u4	Analogový vstupní signál	Double (F64)
iSW	Selektor aktivního signálu	Long (I32)
SW1	Binární vstup pro výběr	Bool
SW2	Binární vstup pro výběr	Bool

Parametr

BINF	Výběr pomocí binárních vstupů	Bool
	off ... zakázáno (výběr přes iSW)	
	on povoleno (výběr přes SW1 a SW2)	

Výstup

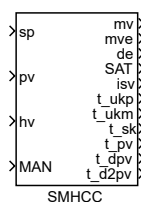
y	Zvolený vstupní signál	Double (F64)

U1..U4	Binární výstup pro selektorovou regulaci	Bool

SMHCC – Regulátor pro procesy s topením a chlazením

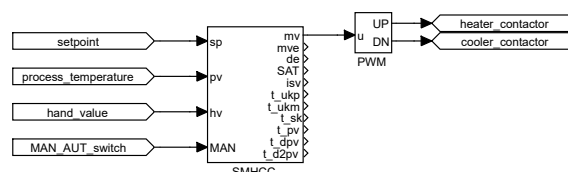
Symbol bloku

Licence: [ADVANCED](#)



Popis funkce

Regulátor **SMHCC** (Sliding Mode Heating/Cooling Controller) je snadno nastavitelný regulátor pro kvalitní regulaci teplotních soustav s dvoustavovým (ON-OFF) topením a dvoustavovým (ON-OFF) chlazením. Klasickým příkladem takových soustav je plastikařský lis. **SMHCC** může být samozřejmě nasazen i na jiné soustavy, kde se dosud běžně používají konvenční termostaty. Pro zajištění správné funkce je nutné blok **SMHCC** doplnit blokem **PWM** (Pulse Width Modulation), jak je patrné z následujícího obrázku.



Blok **SMHCC** pracuje s několika časovými periodami. *První* perioda T_S je vzorkovací perioda měření teploty a je rovněž rovna periodě, se kterou se blok regulátoru **SMHCC** spouští. *Druhá* perioda $T_C = i_{pwm} T_S$ je perioda řízení, se kterou blok **SMHCC** generuje akční zásahy. Tato perioda T_C je totožná s periodou cyklu bloku **PWM**. V každém okamžiku, když se změní akční zásah mv bloku **SMHCC**, algoritmus bloku **PWM** přepočte šířku pulsu a spustí nový **PWM** cyklus. *Třetí* perioda, kterou je třeba stanovit, je perioda spouštění T_R bloku **PWM**. Obecně může být $T_R \neq T_S$. Pro dosažení co nejlepší kvality řízení je doporučeno nastavit periodu T_S na minimální možnou hodnotu (i_{pwm} na maximální možnou hodnotu), poměr T_C/T_S maximální, ale T_C by měla být dostatečně malá vzhledem k dynamice procesu. Pro aplikace v plastikařském průmyslu jsou doporučeny následující hodnoty:

$$T_S = 0.1, \quad i_{pwm} = 100, \quad T_C = 10s, \quad T_R = 0.01s.$$

Zákon řízení bloku regulátoru **SMHCC** v automatickém režimu ($MAN=off$) je založen na diskretní technice dynamického řízení s klouzavým režimem a dále je použit speciální filtr třetího řádu pro odhad první a druhé derivace regulační odchylky.

Po změně požadované hodnoty **sp** (setpoint) se regulátor dostane do první fáze, tzv. *přibližovací*, kdy diskretní proměnná klouzavého režimu

$$s_k \triangleq \ddot{e}_k + 2\xi\Omega\dot{e}_k + \Omega^2 e_k$$

je stlačena do nuly. Ve výše uvedené definici neznámé $e_k, \dot{e}_k, \ddot{e}_k$ po řadě označují filtrovanou regulační odchylku (**pv-sp**), první a druhou derivaci e_k v čase k . Parametry ξ a Ω jsou popsány níže. V druhé fázi (kvazi *klouzavý režim*) je proměnná s_k držena v okolí nulové hodnoty pomocí patřičných zásahů řízení, režim topení se střídá s režimem chlazení. Amplitudy topení a chlazení se adaptují tak, aby se dosáhlo přibližně $s_k = 0$. V důsledku toho je hypotetická spojitá proměnná klouzavého režimu

$$s \triangleq \ddot{e} + 2\xi\Omega\dot{e} + \Omega^2 e$$

stále přibližně nulová. Jinak řečeno regulační odchylka e je popsána diferenciální rovnicí druhého řádu

$$s \triangleq \ddot{e} + 2\xi\Omega\dot{e} + \Omega^2 e = 0.$$

Z toho plyne, že vývoj e může být ovlivněn volbou parametrů ξ a Ω . Poznamenejme, že pro stabilní chování musí být splněno $\xi > 0$ a $\Omega > 0$. Typická optimální hodnota ξ leží v intervalu $[0.1, 8]$. Optimální hodnota Ω je silně závislá na řízeném procesu, pomalejší procesy mají menší hodnotu a rychlejší větší. Doporučená hodnota Ω pro začátek ladění parametrů je $\pi/(5T_C)$.

Řídicí veličina **mv** je obvykle v intervalu $[-1, 1]$. Kladná hodnota odpovídá topení, záporná chlazení, např. **mv** = 1 znamená plné topení. Omezení na **mv** může být zadáno parametry **hilim_p** a **hilim_m**. Omezení může být potřebné, když existuje velká asymetrie mezi topením a chlazením. Jestliže je například chlazení mnohem agresivnější než topení, je vhodné nastavit **hilim_p** = 1 and **hilim_m** < 1. Pokud chceme omezení aplikovat pouze v intervalu po změně požadované hodnoty **sp**, volíme **u0_p** a **u0_m** tak, že platí **u0_p** ≤ **hilim_p** a **u0_m** ≤ **hilim_m**.

Hodnoty amplitud proměnných pro topení a chlazení **t_ukp**, **t_ukm** se automaticky adaptují speciálním algoritmem tak, aby byl dosažen *kvazi klouzavý režim*, ve kterém se střídají znaménka **sk** po každém kroku. V tomto případě se výstup **isv** přepíná mezi 1 a -1. Rychlost adaptace amplitud topení a chlazení je dána časovými konstantami **taup** a **taum**. Obě tyto časové konstanty musí být dostatečně velké, aby zajistily správnou funkci adaptace, ale jejich jemné doladění není nezbytné pro výslednou kvalitu regulace. Pro úplnost dodejme, že **mv** je určena na základě amplitud **t_ukp** a **t_ukm** podle následujícího výrazu

$$if \quad (sk < 0.0) \quad then \quad mv = t_{ukp} \quad else \quad mv = -t_{ukm}.$$

Dále je třeba říci, že dosažení kvazi klouzavého režimu nastává velmi zřídka, protože řízené procesy obsahují dopravní zpoždění a působí na ně poruchy. Vhodným indikátorem kvality „klouzání“ je opět výstup **isv**. Pro jemné doladění je možno v mimořádných případech použít parametr **beta** definující šířku pásma derivačního filtru. Ve většině případů však vyhovuje přednastavená hodnota **beta** = 0.1.

V manuálním režimu (**MAN** = **on**) je vstup regulátoru **hv** kopírován (po případném omezení saturačními mezemi [**hilim_m**, **hilim_p**]) na výstup **mv**. Výstup **mve** poskytuje ekvivalentní amplitudově modulovanou hodnotu akční veličiny **mv** pro informativní účely. Výstup **mve** je získán aplikací filtru prvního řádu s časovou konstantou **tauf** aplikovaným na **mv**.

Tento blok propaguje kvalitu signálu. Více informací je uvedeno v sekci 1.4.

Vstup

sp	Požadovaná hodnota (setpoint)	Double (F64)
pv	Řízená veličina	Double (F64)
hv	Hodnota výstupu v manuálním režimu	Double (F64)
MAN	Manuální nebo automatický režim	Bool
	off ... automatický režim	
	on manuální režim	

Parametr

ipwmc	Délka PWM cyklu (počet vzorkovacích period bloku)	⊙100	Long (I32)
xi	Relativní tlumení	⊙1.0	Double (F64)
om	Přirozená frekvence	⊙0.01	Double (F64)
taup	Časová konstanta adaptace amplitudy topení [s]	⊙700.0	Double (F64)
taum	Časová konstanta adaptace amplitudy chlazení [s]	⊙400.0	Double (F64)
beta	Šířka pásma derivačního filtru	⊙0.01	Double (F64)
hilim_p	Horní saturační mez amplitudy topení	⊙1.0	Double (F64)
hilim_m	Horní saturační mez amplitudy chlazení	⊙1.0	Double (F64)
u0_p	Počáteční hodnota amplitudy topení	⊙1.0	Double (F64)
u0_m	Počáteční hodnota amplitudy chlazení	⊙1.0	Double (F64)
sp_dif	Práh pro detekci změny setpointu	⊙10.0	Double (F64)
tauf	Časová konstanta filtru ekvivalentní akční veličiny	⊙400.0	Double (F64)

Výstup

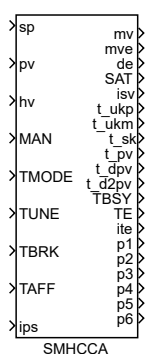
mv	Akční zásah regulátoru (manipulated variable)	Double (F64)
mve	Ekvivalentní akční veličina	Double (F64)
de	Regulační odchylka	Double (F64)
SAT	Saturace	Bool
	off ... lineární zákon řízení	
	on výstup regulátoru je saturován	
isv	Počet kroků přepínací proměnné	Long (I32)
t_ukp	Aktuální amplituda topení	Double (F64)
t_ukm	Aktuální amplituda chlazení	Double (F64)
t_sk	Přepínací proměnná	Double (F64)
t_pv	Filtrovaná řízená veličina	Double (F64)

t_dpv	Filtrovaná první derivace řízené veličiny	Double (F64)
t_d2pv	Filtrovaná druhá derivace řízené veličiny	Double (F64)

SMHCCA – Regulátor pro procesy s topením a chlazením s autotunerem

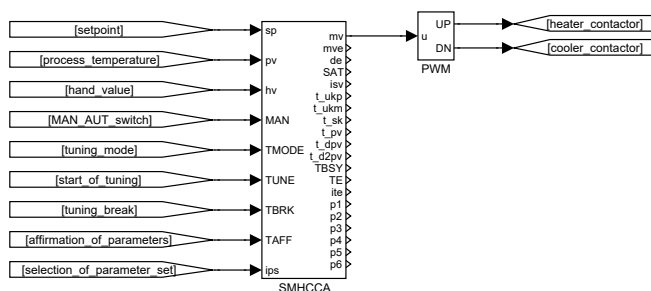
Symbol bloku

Licence: [AUTOTUNING](#)



Popis funkce

Funkční blok **SMHCCA** (Sliding Mode Heating/Cooling Controller with Autotuner) je algoritmem řízení vysoké kvality s vestavěným autotunerem pro automatické ladění parametrů regulátoru. Regulátor je snadno nastavitelný regulátor pro kvalitní regulaci teplotních soustav s dvoustavovým (ON-OFF) topením a dvoustavovým (ON-OFF) chlazením. Klasickým příkladem takových soustav je plastikářský lis. Může být ale samozřejmě nasazen i na jiné soustavy, kde se dosud běžně používají konvenční termostaty. Pro zajištění správné funkce je nutné blok **SMHCCA** doplnit blokem **PWM** (Pulse Width Modulation), jak je patrné z následujícího obrázku.



Principy fungování

Je důležité si uvědomit, že blok **SMHCCA** pracuje s několika časovými periodami. *První* perioda T_S je vzorkovací perioda měřené teploty a je rovněž rovna periodě, se kterou se blok regulátoru **SMHCCA** spouští. *Druhá* perioda $T_C = i_{pwm} T_S$ je perioda řízení, se kterou blok **SMHCCA** generuje akční zásahy. Tato perioda T_C je totožná s periodou cyklu bloku **PWM**. V každém okamžiku, když se změní akční zásah mv bloku **SMHCCA**, algoritmus

bloku **PWM** přepočte šířku pulsu a spustí nový **PWM** cyklus. *Třetí* perioda, kterou je třeba stanovit, je perioda spouštění T_R bloku **PWM**. Obecně může být $T_R \neq T_S$. Pro dosažení co nejlepší kvality řízení je doporučeno nastavit periodu T_S na minimální možnou hodnotu (i_{pwmc} na maximální možnou hodnotu), poměr T_C/T_S maximální, ale T_C by měla být dostatečně malá vzhledem k dynamice procesu. Pro aplikace v plastikářském průmyslu jsou doporučeny následující hodnoty:

$$T_S = 0.1, i_{pwmc} = 50, T_C = 5s, T_R = 0.1s.$$

Všimněte si však, že pro rychlejší řízený systém musí být vzorkovací periody T_S , T_C a T_R zkráceny! Přesněji, pro výběr těchto časových period jsou důležité tři minimální časové konstanty procesu (všechny reálné tepelné procesy mají alespoň tři časové konstanty). Například vzorkovací perioda $T_S = 0.1$ je dostatečně krátká pro takové procesy, které mají alespoň tři časové konstanty, minimální z nich je větší než 10 s a maximální větší než 100 s. Pro správnou funkci regulátoru je nutné, aby tyto časové parametry byly uživatelem vhodně zvoleny podle aktuální dynamiky procesu! Pokud je **SMHCCA** implementován na procesoru s plovoucí desetinnou čárkou, pak je přesné nastavení vzorkovacích period T_S , T_C , T_R a parametru **beta** kritické pro správnou funkci regulátoru. Manuálně zvoleny musí být také některé další parametry s jasným významem popsáným níže. Všechny zbývající parametry (**xi**, **om**, **taup**, **taum**, **tauf**) mohou být automaticky nastaveny vestavěným autotunerem.

Automatický ladící režim

Autotuner pro tento účel používá dvě metody:

- První je určena pro situace, kdy asymetrie procesu není příliš velká (přibližně to znamená, že poměr zisku topení/chlazení nebo chlazení/topení je menší než 5).
- Druhá metoda poskytuje podporu ladění pro silně asymetrické procesy a zatím není implementována (Zatím byla tato metoda vyvinuta a testována pouze v Simulinku).

Navzdory skutečnosti, že první metoda ladění je založena pouze na režimu topení, výsledné parametry jsou obvykle uspokojivé pro oba režimy topení i chlazení kvůli silné robustnosti řízení klouzavým režimem. Ladící procedura je velmi rychlá a může být dokončena během normálního náběhového času teploty procesu od studeného stavu k nastavené hodnotě obvykle bez jakéhokoli zpoždění nebo zhoršení výkonu řízení. Tudiž ladící procedura může být zahrnuta do každého startu od studeného stavu k pracovnímu bodu určenému dostatečně vysokou teplotou.

Nyní bude podrobně popsán implementovaný postup:

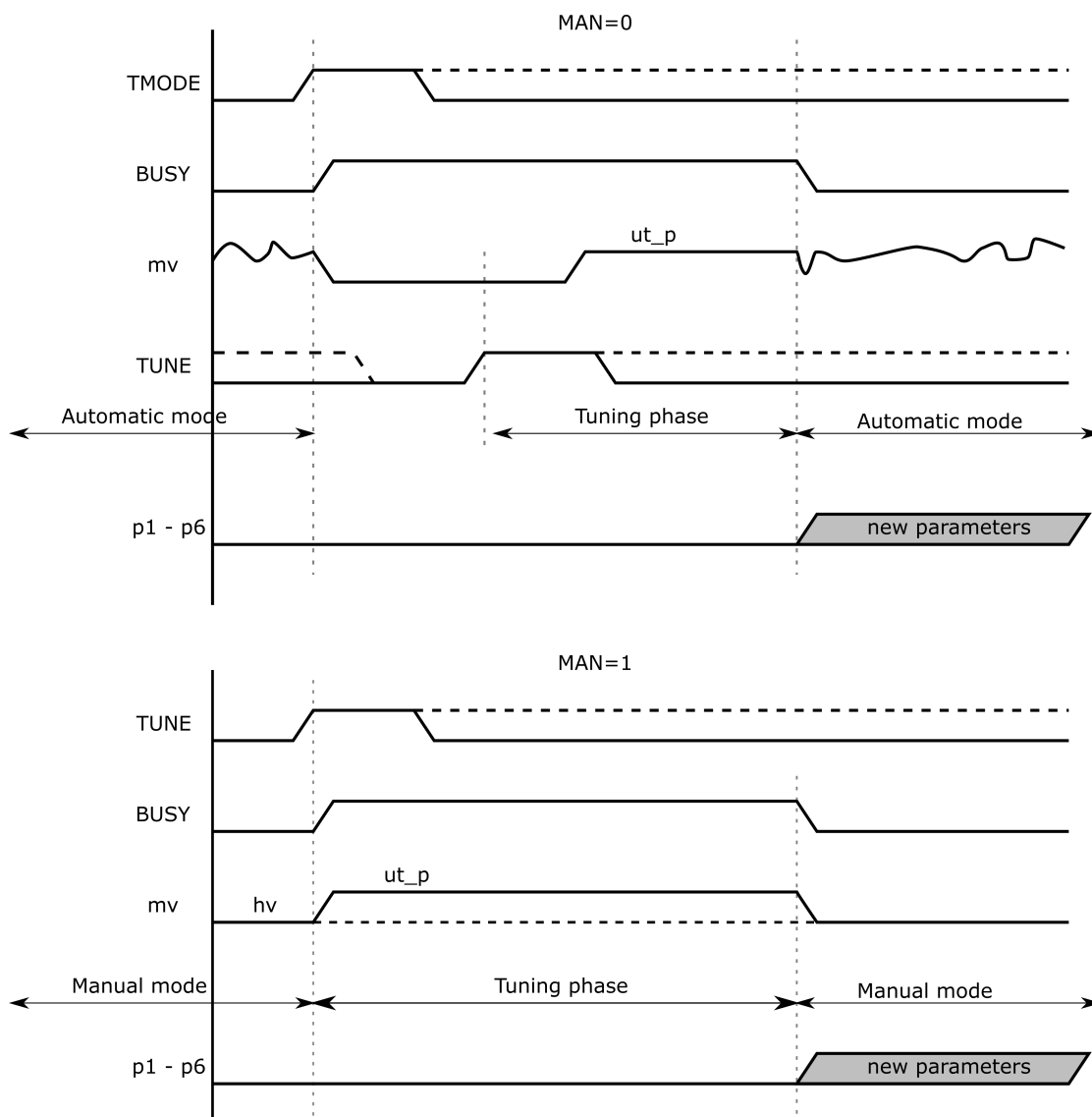
- Ladící procedura začíná v ladícím režimu nebo v manuálním režimu. Pokud je vybrán ladící režim (**TMODE** = **on**), řízená veličina **mv** je automaticky nastavena na nulu a výstup **TBSY** je nastaven na 1 pro indikaci ladící fáze regulátoru. Studený stav procesu je zachován, dokud není na vstupu **TUNE** není indikována vzestupná náběžná hrana **off**→**on**.

- Po nějakém čase (závislém na **beta**), kdy je odhadnuta amplituda šumu, je topení zapnuto s amplitudou danou parametrem **ut_p**. Teplota procesu **pv** a její dvě derivace (výstupy **t_pv**, **t_dpv**, **t_d2pv**) jsou sledovány, aby byly získány optimální parametry regulátoru.
- Pokud ladící procedura skončí bez chyb, pak je **TBSY** nastaven na 0 a regulátor začíná pracovat v manuálním nebo automatickém režimu podle vstupu **MAN**. Pokud **MAN = off** a potvrzovací vstup **TAFF** je nastaven na **on**, pak regulátor začíná pracovat v automatickém režimu s novou sadou parametrů poskytnutou laděním (pokud **TAFF = off**, pak jsou nové parametry pouze zobrazeny na výstupech **p1 . . . p6**).
- Pokud během ladění dojde k nějaké chybě, pak ladící procedura okamžitě zastaví nebo zastaví po splnění podmínky **pv > sp**, výstup **TE** je nastaven na **on** a **ite** indikuje typ chyby. Také v tomto případě regulátor začíná pracovat v režimu určeném vstupem **MAN**. Pokud **MAN = off** pak pracuje v automatickém režimu s původními parametry před laděním!
- Chyby ladění jsou obvykle způsobeny buď nevhodným nastavením parametru **beta** nebo příliš nízkou hodnotou **sp**. Vhodná hodnota **beta** se pohybuje v intervalu (0.001,0.1). Pokud je drift a šum v **pv** velký, musí být zvolena malá hodnota **beta** zejména pro fázi ladění. Výchozí hodnota (**beta=0.01**) by měla dobře fungovat pro aplikace plastikářského lisu. Správná hodnota dává řádně filtrovaný signál druhé derivace teploty procesu **t_d2pv**. Tento dobře filtrovaný signál (odpovídající nízké hodnotě **beta**) je hlavně nutný pro správné ladění. Pro řízení může být parametr **beta** někdy mírně zvýšen.
- Ladící procedura může být také zahájena z manuálního režimu (**MAN = off**) s jakoukoli konstantní hodnotou vstupu **hv**. V tomto případě musí být však zajištěn ustálený stav. Opět, ladění je zahájeno náběžnou hranou **off→on** na vstupu **TUNE** a po zastavení ladění regulátor pokračuje v manuálním režimu. V obou případech se výsledné parametry objeví na výstupech **p1, . . . , p6**.

Pro jednotlivé hodnoty **ips** mají parametry **p1, . . . , p6** následující významy:

- **0:** Parametry regulátoru
 - **p1** . . . doporučená perioda řízení T_C
 - **p2** . . . **xi**
 - **p3** . . . **om**
 - **p4** . . . **taup**
 - **p5** . . . **taum**
 - **p6** . . . **tauf**
- **1:** Pomocné parametry
 - **p1** . . . **htp2** – čas vrcholu ve druhé derivaci **pv**

- p2 ... hpeak2 – vrcholová hodnota ve druhé derivaci pv
- p3 ... d2 – amplituda vrchol k vrcholu t_d2pv
- p4 ... tgain



Automatický režim

Zákon řízení bloku regulátoru SMHCCA v automatickém režimu (**MAN=off**) je založen na diskretní technice dynamického řízení s klouzavým režimem a dále je použit speciální filtr třetího řádu pro odhad první a druhé derivace regulační odchylky.

Po změně požadované hodnoty **sp** (setpoint) se regulátor dostane do první fáze, tzv.

přibližovací, kdy diskrétní proměnná klouzavého režimu

$$s_k \stackrel{\Delta}{=} \ddot{e}_k + 2\xi\Omega\dot{e}_k + \Omega^2 e_k$$

je stlačena do nuly. Ve výše uvedené definici neznámé $e_k, \dot{e}_k, \ddot{e}_k$ po řadě označují filtrovanou regulační odchylku (**pv-sp**), první a druhou derivaci e_k v čase k . Parametry ξ a Ω jsou popsány níže. V druhé fázi (kvazi *klouzavý režim*) je proměnná s_k držena v okolí nulové hodnoty pomocí patřičných zásahů řízení, režim topení se střídá s režimem chlazení. Amplitudy topení a chlazení se adaptují tak, aby se dosáhlo přibližně $s_k = 0$. V důsledku toho je hypotetická spojitá proměnná klouzavého režimu

$$s \stackrel{\Delta}{=} \ddot{e} + 2\xi\Omega\dot{e} + \Omega^2 e$$

stále přibližně nulová. Jinak řečeno regulační odchylka e je popsána diferenciální rovnicí druhého řádu

$$s \stackrel{\Delta}{=} \ddot{e} + 2\xi\Omega\dot{e} + \Omega^2 e = 0.$$

Z toho plyne, že vývoj e může být ovlivněn volbou parametrů ξ a Ω . Poznamenejme, že pro stabilní chování musí být splněno $\xi > 0$ a $\Omega > 0$. Typická optimální hodnota ξ leží v intervalu $[0.1, 8]$. Optimální hodnota Ω je silně závislá na řízeném procesu, pomalejší procesy mají menší hodnotu a rychlejší větší. Doporučená hodnota Ω pro začátek ladění parametrů je $\pi/(5T_C)$.

Řídicí veličina mv je obvykle v intervalu $[-1, 1]$. Kladná hodnota odpovídá topení, záporná chlazení, např. $mv = 1$ znamená plné topení. Omezení na mv může být zadáno parametry **hilim_p** a **hilim_m**. Omezení může být potřebné, když existuje velká asymetrie mezi topením a chlazením. Jestliže je například chlazení mnohem agresivnější než topení, je vhodné nastavit **hilim_p** = 1 and **hilim_m** < 1. Pokud chceme omezení aplikovat pouze v intervalu po změně požadované hodnoty **sp**, volíme **u0_p** a **u0_m** tak, že platí **u0_p** ≤ **hilim_p** a **u0_m** ≤ **hilim_m**.

Hodnoty amplitud proměnných pro topení a chlazení **t_ukp**, **t_ukm** se automaticky adaptují speciálním algoritmem tak, aby byl dosažen *kvazi klouzavý režim*, ve kterém se střídají znaménka **sk** po každém kroku. V tomto případě se výstup **isv** přepíná mezi 1 a -1. Rychlost adaptace amplitud topení a chlazení je dána časovými konstantami **taup** a **taum**. Obě tyto časové konstanty musí být dostatečně velké, aby zajistily správnou funkci adaptace, ale jejich jemné doladění není nezbytné pro výslednou kvalitu regulace. Pro úplnost dodejme, že mv je určena na základě amplitud **t_ukp** a **t_ukm** podle následujícího výrazu

$$if \ (sk < 0.0) \ then \ mv = t\_ukp \ else \ mv = -t\_ukm .$$

Dále je třeba říci, že dosažení kvazi klouzavého režimu nastává velmi zřídka, protože řízené procesy obsahují dopravní zpoždění a působí na ně poruchy. Vhodným indikátorem kvality „klouzání“ je opět výstup **isv**. Pro jemné doladění je možno v mimořádných případech použít parametr **beta** definující šířku pásma derivačního filtru. Ve většině případů však vyhovuje přednastavená hodnota **beta** = 0.1.

Manuální režim

V manuálním režimu ($MAN = on$) je vstup regulátoru h_v kopírován (po případném omezení saturačními mezemi $[-hilim\_m, hilim\_p]$) na výstup mv . Výstup mve poskytuje ekvivalentní amplitudově modulovanou hodnotu akční veličiny mv pro informativní účely. Výstup mve je získán aplikací filtru prvního řádu s časovou konstantou $tauf$ aplikovaným na mv .

Tento blok propaguje kvalitu signálu. Více informací je uvedeno v sekci 1.4.

Vstup

sp	Požadovaná hodnota (setpoint)	Double (F64)
pv	Řízená veličina	Double (F64)
h_v	Hodnota výstupu v manuálním režimu	Double (F64)
MAN	Manuální nebo automatický režim	Bool
	$off \dots$ automatický režim	
	$on \dots$ manuální režim	
$TMODE$	Režim ladění	Bool
$TUNE$	Zahájení ladicího experimentu	Bool
$TBRK$	Ukončení ladicího experimentu	Bool
$TAFF$	Přijetí výsledků ladicího experimentu	Bool
	$off \dots$ parametry jsou pouze vypočítány	
	$on \dots$ parametry jsou dosazeny do řídicího algoritmu	
ips	Význam výstupních signálů	Long (I32)
	$0 \dots$ parametry regulátoru	
	$1 \dots$ pomocné parametry	

Parametr

$ipwmc$	Délka PWM cyklu (počet vzorkovacích period bloku)	$\odot 100$	Long (I32)
xi	Relativní tlumení	$\downarrow 0.5 \uparrow 8.0 \odot 1.0$	Double (F64)
om	Přirozená frekvence	$\downarrow 0.0 \odot 0.01$	Double (F64)
$taup$	Časová konstanta adaptace amplitudy topení [s]	$\odot 700.0$	Double (F64)
$taum$	Časová konstanta adaptace amplitudy chlazení [s]	$\odot 400.0$	Double (F64)
$beta$	Šířka pásma derivačního filtru	$\odot 0.01$	Double (F64)
$hilim\_p$	Horní saturační mez amplitudy topení	$\downarrow 0.0 \uparrow 1.0 \odot 1.0$	Double (F64)
$hilim\_m$	Horní saturační mez amplitudy chlazení	$\downarrow 0.0 \uparrow 1.0 \odot 1.0$	Double (F64)
$u0\_p$	Počáteční hodnota amplitudy topení	$\odot 1.0$	Double (F64)
$u0\_m$	Počáteční hodnota amplitudy chlazení	$\odot 1.0$	Double (F64)
sp_dif	Práh pro detekci změny setpointu	$\odot 10.0$	Double (F64)
$tauf$	Časová konstanta filtru ekvivalentní akční veličiny	$\odot 400.0$	Double (F64)
itm	Metoda ladění regulátoru	$\odot 1$	Long (I32)
	$1 \dots$ omezeno na symetrické procesy		
	$2 \dots$ asymetrické procesy (zatím není implementováno)		
ut_p	Amplituda topení pro ladicí experiment	$\downarrow 0.0 \uparrow 1.0 \odot 1.0$	Double (F64)
ut_m	Amplituda chlazení pro ladicí experiment	$\downarrow 0.0 \uparrow 1.0 \odot 1.0$	Double (F64)

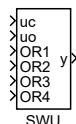
Výstup

mv	Akční zásah regulátoru (manipulated variable)	Double (F64)
mve	Ekvivalentní akční veličina	Double (F64)
de	Regulační odchylka	Double (F64)
SAT	Saturace	Bool
	off ... lineární zákon řízení	
	on výstup regulátoru je saturován	
isv	Počet kroků přepínací proměnné	Long (I32)
t_ukp	Aktuální amplituda topení	Double (F64)
t_ukm	Aktuální amplituda chlazení	Double (F64)
t_sk	Přepínací proměnná	Double (F64)
t_pv	Filtrovaná řízená veličina	Double (F64)
t_dpv	Filtrovaná první derivace řízené veličiny	Double (F64)
t_d2pv	Filtrovaná druhá derivace řízené veličiny	Double (F64)
TBSY	Příznak probíhajícího ladicího experimentu	Bool
TE	Příznak chyby během ladění	Bool
	off ... ladění proběhlo bez chyby	
	on během ladění se vyskytla chyba	
ite	Kód chyby	Long (I32)
	0 bez chyby	
	1 příliš zašuměné pv, zkontroluj teplotní vstup 2	
	2 nesprávný parametr ut_p	
	3 setpoint je příliš nízký	
	4 vzorkovací perioda je příliš nízká nebo je druhá derivace příliš zašuměná	
	5 předčasné ukončení ladicího experimentu	
p1..p6	Výsledky identifikace a návrhu regulátoru	Double (F64)

SWU – Přepínač vstupu pro vysledování

Symbol bloku

Licence: [STANDARD](#)



Popis funkce

Blok SWU je určen pro přepínání vhodného signálu na vstup pro vysledování bloků [PIDU](#) a [MCU](#). V případě, že všechny vstupy OR1, ..., OR4 jsou logické nuly (**off**), potom na výstup y je kopírována hodnota vstupu uc, v opačném případě hodnota vstupu uo.

Tento blok propaguje kvalitu signálu. Více informací je uvedeno v sekci [1.4](#).

Vstup

uc	Signál platný pokud všechny OR _i =0	Double (F64)
uo	Signál platný pokud alespoň jeden OR _i =1	Double (F64)
OR1..OR4	Logický výstupní signál	Bool

Výstup

y	Analogový výstupní signál	Double (F64)

TSE – Třístavový prvek

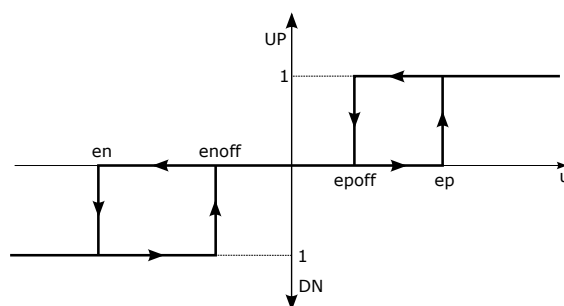
Symbol bloku

Licence: [STANDARD](#)



Popis funkce

Blok TSE transformuje analogový vstup u na třístavový výstup (méně, nečinnost, více) podle níže uvedeného obrázku.



Tento blok propaguje kvalitu signálu. Více informací je uvedeno v sekci [1.4](#).

Vstup

u	Analogový vstupní signál	Double (F64)

Parametr

ep	Hodnota pro přepnutí do režimu UP (nahoru)	$\odot 1.0$ Double (F64)
en	Hodnota pro přepnutí do režimu DN (dolů)	$\odot -1.0$ Double (F64)
$epoff$	Hodnota pro vypnutí režimu UP (nahoru)	$\odot 0.5$ Double (F64)
$enoff$	Hodnota pro vypnutí režimu DN (dolů)	$\odot -0.5$ Double (F64)

Výstup

UP	Signál UP (nahoru, více)	Bool
DN	Signál DN (dolů, méně)	Bool

Kapitola 8

LOGIC – Logické řízení

Obsah

AND – Logický součin dvou signálů	302
ANDQUAD, ANDOCT, ANDHEXD – Logický součin více signálů	303
ATMT – Automat pro sekvenční řízení	304
BDOCT, BDHEXD – Bitové demultiplexery	306
BITOP – Bitová operace dvou celočíselných signálů	307
BMOCT, BMHEXD – Bitové multiplexery	308
COUNT – Řízený čítač	309
EATMT – Rozšířený automat pro sekvenční řízení	311
EDGE – Detekce hrany logického signálu	314
EQ – Shodnost dvou signálů	315
INTSM – Bitový posun a maska nad celým číslem	316
ISSW – Jednoduchý přepínač celočíselných signálů	317
ITOI – Transformace celých a binárních čísel	318
NOT – Logická negace	319
OR – Logický součet dvou signálů	320
ORQUAD, OROCT, ORHEXD – Logický součet více signálů	321
RS – Klopný obvod	322
SR – Klopný obvod	323
TIMER – Vícefunkční časovač	324

Knihovna LOGIC zahrnuje řadu bloků pro provádění logických a sekvenčních operací. Zahrnuje základní booleovské bloky jako **AND**, **OR**, **NOT** pro základní logické operace a pokročilé bloky jako **ATMT** pro konečné automaty. Bloky jako **COUNT** a **TIMER** rozšiřují funkčnost na obousměrné počítání pulsů a operace založené na čase. Další prvky jako **BITOP**, **BMOCT** a **BDOCT** nabízejí bitové operace a možnosti multiplexování/demultiplexování, čímž zvyšují všestrannost knihovny při ovládání kombinační a sekvenční logiky.

AND – Logický součin dvou signálů

Symbol bloku

Licence: [STANDARD](#)



Popis funkce

Blok AND dělá logický součin dvou vstupních signálů $Y = U1 \wedge U2$. Pro logický součin více vstupních signálů použijte jeden z bloků [ANDQUAD](#), [ANDOCT](#) nebo [ANDHEXD](#).

Tento blok propaguje kvalitu signálu. Více informací je uvedeno v sekci [1.4](#).

Vstup

U1	První logický vstup bloku	Bool
U2	Druhý logický vstup bloku	Bool

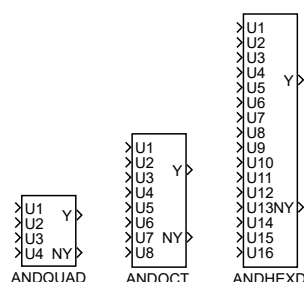
Výstup

Y	Výstup bloku, logický součin	Bool
NY	Negace výstupního signálu Y	Bool

ANDQUAD, ANDOCT, ANDHEXD – Logický součin více signálů

Symboly bloků

Licence: [STANDARD](#)



Popis funkce

Bloky ANDQUAD, ANDOCT a ANDHEXD vyhodnocují logický součin až 16 vstupních signálů U1 až U16. Signály, jejichž seznam je uveden v parametru n1 se před provedením logického součinu negují.

Pokud je tedy parametr n1 prázdný, tak se provádí logický součin $Y = U1 \wedge U2 \wedge U3 \wedge U4 \wedge U5 \wedge U6 \wedge U7 \wedge U8$. Pokud bude například $n1=1,3..5$, pak $Y = \neg U1 \wedge U2 \wedge \neg U3 \wedge \neg U4 \wedge \neg U5 \wedge U6 \wedge \dots \wedge U16$.

Pokud pracujete s méně než 8 signály, je potřeba ošetřit nepřípojené vstupy bloku pomocí parametru n1. Pokud pracujete pouze se dvěma vstupními signály, zvažte použití bloku [AND](#).

Tento blok propaguje kvalitu signálu. Více informací je uvedeno v sekci [1.4](#).

Vstup

U1..U16	Logický vstupní signál	Bool

Parametr

n1	Seznam negovaných signálů	Long (I32)

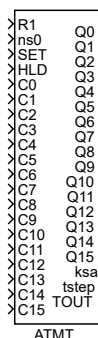
Výstup

Y	Výstup bloku, logický součin	Bool
NY	Negace výstupního signálu Y	Bool

ATMT – Automat pro sekvenční řízení

Symbol bloku

Licence: [STANDARD](#)



Popis funkce

Blok ATMT realizuje konečný automat až s 16 stavy a 16 podmínkami přechodů mezi nimi.

Aktuální stav automatu i , $i = 0, 1, \dots, 15$, je indikován pomocí binárních výstupů $Q0, Q1, \dots, Q15$. Pokud je automat ve stavu i , je nastaven příslušný výstup $Q_i = \text{on}$. Aktuální stav automatu je též indikován celočíselným výstupem $\text{ksa} \in \{0, 1, \dots, 15\}$.

Podmínky přechodů C_k , $k = 0, 1, \dots, 15$ jsou aktivovány pomocí binárních vstupů bloku $C0, C1, \dots, C15$. Pokud je $C_k = \text{on}$, je podmínka C_k splněna, naopak pro $C_k = \text{off}$ splněna není.

Funkce automatu se zadává pomocí tabulky stavů a přechodů:

$S1$	$C1$	$NS1$
$S2$	$C2$	$NS2$
		$\dots$
S_n	C_n	NS_n

Každý řádek této tabulky vyjadřuje jedno pravidlo přechodu. Např. první řádek

$S1$	$C1$	$NS1$

má tento význam:

Jestliže (aktuální stav je $S1$ AND podmínka přechodu $C1$ je splněna),
potom přejdi do následujícího stavu $NS1$.

Výše uvedenou tabulku lze získat ze stavového diagramu automatu nebo z popisu automatu v jazyce SFC (Sequential Function Charts, dříve Grafcet).

Vstup $R1 = \text{on}$ resetuje stav automatu do počátečního stavu $S0$, přičemž vstup $R1$ má prioritu před vstupem SET . Náběžná hrana na vstupu SET způsobí přechod z aktuálního

stavu do stavu **ns0**. Vstup **HLD = on** zablokuje činnost automatu, tzn. automat setrvává v daném stavu i v případě, že je splněna některá podmínka přechodu, rovněž je zastaveno inkrementování času **tstep** a generování výstupu **TOUT**. Výstup **TOUT** indikuje, že automat setrval v daném stavu déle, než je povoleno. Časová omezení **TOi** jednotlivých stavů se definují pomocí vektoru **touts**. Pokud je **TOi = 0**, není pro daný stav nastaveno žádné časové omezení. Výstup **TOUT** je automaticky nastavován na hodnotu **off** při každém přechodu mezi stavy automatu.

Pomocí parametru **moresteps** lze povolit přechod automatu o více kroků v jednom cyklu. Tuto možnost je však vždy potřeba pečlivě zvážit, zejména při použití výstupu **TOUT** v podmínkách pro přechod do dalších stavů. V takovém případě je vhodné konstruovat podmínku přechodu nejen pomocí výstupu **TOUT**, ale zahrnout do ní i informaci o stavu automatu **ksa**.

Součástí systému **REXYGEN** je také program **SFCeditor**, který umožňuje tvorbu SFC schémat v grafickém návrhovém prostředí. Editor se spouští z programu **REXYGEN Studio** kliknutím na tlačítko *Configure* na kartě parametrů bloku **ATMT**. Uživatelská příručka editoru je k dispozici jako samostatný dokument.

Tento blok propaguje kvalitu signálu. Více informací je uvedeno v sekci 1.4.

Vstup

R1	Reset bloku	Bool
ns0	Cílový stav pro přechod vynucený vstupem SET	Long (I32)
SET	Vynucení přechodu do stavu ns0	Bool
HLD	Pozastavení	Bool
C0..C15	Podmínka přechodu	Bool

Parametr

moresteps	Povolit více přechodů mezi stavy v jednom cyklu off ... zakázáno on povoleno	Bool
sfcname	Jméno souboru pro ukládání dat speciálního editoru	String
STT	Tabulka přechodů mezi stavy ⊙[0 0 1; 1 1 2; 2 2 3; 3 3 0]	Byte (U8)
touts	Časové limity pro jednotlivé stavy ⊙[1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16]	Double (F64)

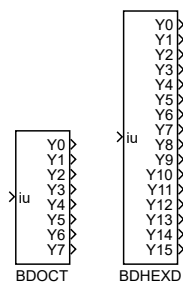
Výstup

Q0..Q15	Indikátor aktivního stavu	Bool
ksa	Celočíselná reprezentace stavu	Long (I32)
tstep	Čas uplynulý od posledního přechodu mezi stavy	Double (F64)
TOUT	Příznak překročení časového limitu pro aktuální stav	Bool

BDOCT, BDHEXD – Bitové demultiplexery

Symboly bloků

Licence: [STANDARD](#)



Popis funkce

Bloky **BDOCT** a **BDHEXD** pracují jako bitové demultiplexery a lze je výhodně použít pro rozebírání celočíselných signálů na jednotlivé bity. Oba bloky se od sebe liší jen počtem výstupů, blok **BDOCT** má 8 bitových výstupů, blok **BDHEXD** jich má 16. Výstupní signály Y_i jsou přímo tvořeny bity signálu, který vznikne bitovým posunem vstupu iu o $shift$ bitů vpravo, přičemž v signálu Y_0 je vždy nejnížší bit tohoto čísla.

Tento blok propaguje kvalitu signálu. Více informací je uvedeno v sekci [1.4](#).

Vstup

iu	Vstupní signál k dekompozici	Long (I32)

Parametr

shift	Bitový posun vstupního signálu	↓0 ↑31	Long (I32)
vtype	Číselný typ	⊙4	Long (I32)
	2 Byte (U8)		
	3 Short (I16)		
	4 Long (I32)		
	5 Word (U16)		
	6 DWord (U32)		
	10 Large (I64)		

Výstup

$Y_0 \dots Y_{15}$	Jednotlivý bit vstupního signálu	Bool

BITOP – Bitová operace dvou celočíselných signálů

Symbol bloku

Licence: [STANDARD](#)



Popis funkce

Blok **BITOP** provádí operaci $i1 \circ i2$ na vstupních signálech po jednotlivých bitech. Výsledkem je celočíselný výstup n . Kód zvolené bitové operace je uveden v parametru **iop** popsaném níže. V případě bitové negace a dvojkových doplňků se operace provádí pouze se vstupem $i1$ (tj. operace je unární).

Tento blok propaguje kvalitu signálu. Více informací je uvedeno v sekci [1.4](#).

Vstup

i1	První celočíselný vstup bloku	Long (I32)
i2	Druhý celočíselný vstup bloku	Long (I32)

Parametr

iop	Bitová operace	⊙1 Long (I32)
	1 bitové NOT	
	2 bitové OR	
	3 bitové AND	
	4 bitové XOR	
	5 posun vlevo	
	6 posun vpravo	
	7 dvojkový doplněk signálu - byte	
	8 dvojkový doplněk signálu - word	
	9 dvojkový doplněk signálu - long	
vtype	Číselný typ	⊙4 Long (I32)
	2 Byte (U8)	
	3 Short (I16)	
	4 Long (I32)	
	5 Word (U16)	
	6 DWord (U32)	
	10 Large (I64)	

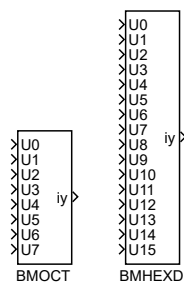
Výstup

n	Výsledek bitové operace	Long (I32)

BMOCT, BMHEXD – Bitové multiplexery

Symboly bloků

Licence: [STANDARD](#)



Popis funkce

Bloky **BMOCT** a **BMHEXD** pracují jako bitové multiplexery a lze je výhodně využít ke skládání celočíselných signálů z jednotlivých bitů. Oba bloky se od sebe liší jen počtem vstupů, blok **BMOCT** má 8 bitových vstupů, blok **BMHEXD** jich má 16. V případě, že parametr **shift** = 0, jsou jednotlivé bity výstupního signálu **iy** přímo tvořeny vstupními signály **Ui**. V nejnižším bitu je vždy signál **U0**.

Tento blok propaguje kvalitu signálu. Více informací je uvedeno v sekci [1.4](#).

Vstup

U0..U15 Jednotlivý bit výstupního signálu Bool

Parametr

shift	Bitový posun výstupního signálu	↓0 ↑31	Long (I32)
vtype	Číselný typ	⊙4	Long (I32)
	2 Byte (U8)		
	3 Short (I16)		
	4 Long (I32)		
	5 Word (U16)		
	6 DWord (U32)		
	10 Large (I64)		

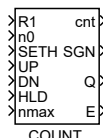
Výstup

iy Výsledný výstupní signál Long (I32)

COUNT – Řízený čítač

Symbol bloku

Licence: [STANDARD](#)



Popis funkce

Blok COUNT je určen pro obousměrné čítání pulsů – přesněji náběžných hran na vstupech UP a DN. Při výskytu náběžné hrany na vstupu UP (DN) se výstup **cnt** zvětší o 1 (sníží o 1). Současný výskyt náběžných hran na vstupech UP a DN indikuje výstup E jako chybu čítání. Resetování výstupu **cnt** na hodnotu 0 lze provést vstupem R1 (pokud je R1 = on je výstup **cnt** držen na hodnotě 0). Nastavení výstupu **cnt** na hodnotu n0 zajistí vstup SETH = on (pokud SETH = on je výstup **cnt** držen na hodnotě n0). Vstup R1 má vyšší prioritu než vstup SETH. Vstup HLD = on způsobí zastavení čítání. Stav čítače $\text{cnt} \geq \text{nmax}$ způsobí nastavení výstupu Q na hodnotu on.

Tento blok propaguje kvalitu signálu. Více informací je uvedeno v sekci [1.4](#).

Vstup

R1	Reset bloku	Bool
n0	Hodnota pro nastavení čítače	Long (I32)
SETH	Nastavení hodnoty čítače	Bool
UP	Vstup pro přičítání	Bool
DN	Vstup pro odečítání	Bool
HLD	Zmrazení čítače	Bool
	off ... čítač běží	
	on čítač je zablokován	
nmax	Cílová hodnota čítače	Long (I32)

Výstup

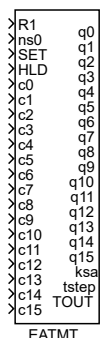
cnt	Celkový počet načtených pulzů	Long (I32)
SGN	Znaménko výstupu cnt	Bool
	off ... menší nebo rovno nule	
	on větší než nula	
Q	Indikátor dosažení cílové hodnoty	Bool
	off ... cílová hodnota nebyla dosažena	
	on dosažena cílová hodnota	

E	Příznak chyby	Bool
	off ... bez chyby	
	on nastala chyba	

EATMT – Rozšířený automat pro sekvenční řízení

Symbol bloku

Licence: [ADVANCED](#)



Popis funkce

Blok EATMT implementuje konečný automat s maximálně 256 stavy a 256 pravidly pro přechody, čímž rozšiřuje možnosti bloku [ATMT](#).

Současný stav automatu i , $i = 0, 1, \dots, 255$, je indikován jednotlivými bity celočíselných výstupů $q0, q1, \dots, q15$. Pouze jeden bit s indexem $i, \text{MOD}, 16$ na výstupu $q(i, \text{DIV}, 16)$ je nastaven na 1. Ostatní bity tohoto výstupu a ostatní výstupy jsou nulové. Bity jsou číslovány od nuly, začínající nejméně významným bitem. Všimněte si, že operátory DIV a MOD označují celočíselné dělení a zbytek po celočíselném dělení. Současný stav je také indikován výstupem $ksa \in 0, 1, \dots, 255$.

Podmínky pro přechod C_k , $k = 0, 1, \dots, 255$, jsou aktivovány jednotlivými bity vstupů $c0, c1, \dots, c15$. k -tá podmínka pro přechod je splněna, když je $(k, \text{MOD}, 16)$ -tý bit vstupu $c(k, \text{DIV}, 16)$ roven 1. Přechod jinak nemůže proběhnout.

Pro kompozici vstupních signálů $c0, c1, \dots, c15$ z jednotlivých Booleovských signálů mohou být použity bitové multiplexory [BMHEXD](#) nebo [BMOCT](#). Podobně mohou být výstupní signály $q0, q1, \dots, q15$ dekomponovány pomocí bitových demultiplexorů [BDHEXD](#) nebo [BDOCT](#).

Funkce automatu je definována následující tabulkou přechodů:

$S1$	$C1$	$FS1$
$S2$	$C2$	$FS2$
		$\dots$
S_n	C_n	FS_n

Každý řádek této tabulky vyjadřuje jedno pravidlo přechodu. Např. první řádek

$S1 \quad C1 \quad FS1$

má tento význam:

Jestliže (aktuální stav je $S1$ AND podmínka přechodu $C1$ je splněna),
potom přejdi do následujícího stavu $FS1$.

Výše popsaný význam řádku tabulky platí pro $C1 < 1000$. Pro $C1 \geq 1000$ se předpokládá negace $(C1 - 1000)$ -té podmínky pro přechod.

Výše zmíněnou tabulku lze snadno sestavit z diagramu stavů automatu nebo z popisu SFC (Sequential Function Charts, dříve Grafcet).

Vstup $R1 = \text{on}$ resetuje automat do počátečního stavu $S0$. Vstup SET umožňuje manuální přechod ze současného stavu do stavu $ns0$ při detekci náběžné hrany. Vstup $R1$ má přednost před vstupem SET . Vstup $HLD = \text{on}$ zastaví vykonávání automatu, takže zůstává ve svém současném stavu bez ohledu na vstupní signály ci a časovač $tstep$ se nezvyšuje. Výstup $TOUT$ signalizuje, že stroj zůstává v daném stavu déle, než se očekávalo. Časové limity TOi pro jednotlivé stavy jsou definovány polem $touts$. Pokud je TOi nastaveno na nulu, pro daný stav neexistuje časový limit. Výstup $TOUT$ je nastaven na off vždy, když automat změní svůj stav.

Je možné umožnit více přechodů stavů v jednom cyklu pomocí parametru $moresteps$. Tato možnost by však měla být pečlivě zvážena a otestována, zejména pokud je výstup $TOUT$ použit v podmínkách pro přechod. V takovém případě se silně doporučuje zahrnout do podmínek pro přechod také výstup ksa .

Vývojové nástroje REXYGEN zahrnují také program $SFCEditor$. Pomocí tohoto nástroje můžete graficky vytvářet schémata SFC. Tento editor spustíte z REXYGEN Studio kliknutím na tlačítko *Konfigurovat* v dialogu parametrů bloku $EATMT$.

Tento blok propaguje kvalitu signálu. Více informací je uvedeno v sekci 1.4.

Vstup

$R1$	Reset bloku	Bool
$ns0$	Cílový stav pro přechod vynucený vstupem SET	Long (I32)
SET	Vynucení přechodu do stavu $ns0$	Bool
HLD	Pozastavení	Bool
$c0..c15$	Podmínka přechodu	Long (I32)

Parametr

$moresteps$	Povolit více přechodů mezi stavy v jednom cyklu off ... zakázáno on povoleno	Bool
$sfcname$	Jméno souboru pro ukládání dat speciálního editoru	String
STT	Tabulka přechodů mezi stavy $\odot[0\ 0\ 1; 1\ 1\ 2; 2\ 2\ 3; 3\ 3\ 0]$	Short (I16)
$touts$	Časové limity pro jednotlivé stavy $\odot[1\ 2\ 3\ 4\ 5\ 6\ 7\ 8\ 9\ 10\ 11\ 12\ 13\ 14\ 15\ 16]$	Double (F64)

Výstup

q0..q15	Indikátor aktivního stavu	Long (I32)
ksa	Celočíselná reprezentace stavu	Long (I32)
tstep	Čas uplynulý od posledního přechodu mezi stavy	Double (F64)
TOUT	Příznak překročení časového limitu pro aktuální stav	Bool

EDGE – Detekce hrany logického signálu

Symbol bloku

Licence: [STANDARD](#)



Popis funkce

Blok **EDGE** detekuje na vstupním signálu **U** náběžnou (**off**→**on**), sestupnou (**on**→**off**) nebo obě uvedené hrany podle hodnoty parametru **iedge**. V případě nalezení požadované hrany (změny vstupního signálu) je na jeden krok nastavena hodnota výstupu **Y** na **on**. Po dobu, kdy se hodnota vstupního signálu nemění je hodnota výstupu **Y** rovna **off**. Hodnota výstupu **Y** zůstane nulová i v případě, že v parametru **iedge** je zvolena náběžná (sestupná) hrana a v signálu se vyskytne hrana opačná, tj. sestupná (náběžná).

Tento blok propaguje kvalitu signálu. Více informací je uvedeno v sekci [1.4](#).

Vstup

U	Logický vstupní signál	Bool

Parametr

iedge	Typ detekovaných hran	⊙1 Long (I32)
	1 náběžná hrana	
	2 sestupná hrana	
	3 obě hrany	

Výstup

Y	Logický výstupní signál	Bool

EQ – Shodnost dvou signálů

Symbol bloku

Licence: [STANDARD](#)



Popis funkce

Blok porovnává dva vstupní signály. Výstup $Y = \text{on}$ je nastaven, pokud mají oba signály stejnou hodnotu. Oba signály musí být shodně buď číselného typu nebo řetězce. Převod mezi numerickými typy je zajištěn: například hodnoty 2,0 (double) a 2 (long) jsou vyhodnoceny jako ekvivalentní. Porovnávání matic nebo jiných komplexních typů není podporováno.

Tento blok propaguje kvalitu signálu. Více informací je uvedeno v sekci [1.4](#).

Vstup

u1	Vstup bloku	Any
u2	Vstup bloku	Any

Výstup

Y	Logický výstupní signál	Bool
NY	Negace výstupního signálu Y	Bool

INTSM – Bitový posun a maska nad celým číslem

Symbol bloku

Licence: [STANDARD](#)



Popis funkce

Blok INTSM provádí bitový posun vstupního čísla *i* o *shift* bitů doprava (pro kladný *shift*) nebo doleva (záporný *shift*). Volné bity vzniklé posunem jsou vyplněny nulami.

Výstupní hodnota *n* je logickým součinem (AND) bitově posunutého vstupu *i* a bitové masky *mask*.

Typické využití bloku spočívá v extrakci hodnoty jednoho nebo více sousedních bitů z určité pozice v celočíselném registru vyčteném z externího systému.

Tento blok propaguje kvalitu signálu. Více informací je uvedeno v sekci [1.4](#).

Vstup

<i>i</i>	Celočíselný signál pro zpracování	Large (I64)
	↓-9.22337E+18 ↑9.22337E+18	

Parametr

<i>shift</i>	Bitový posun (záporné číslo=doleva, kladné číslo=doprava)	Long (I32)
	↓-63 ↑63	
<i>mask</i>	Bitová maska (aplikovaná po bitovém posunu)	Large (I64)
	↓0 ↑4294970000 ⊕4294967295	
<i>vtype</i>	Číselný typ	⊕4 Long (I32)
	2 Byte (U8)	
	3 Short (I16)	
	4 Long (I32)	
	5 Word (U16)	
	6 DWord (U32)	
	10 Large (I64)	

Výstup

<i>n</i>	Výsledná celočíselná hodnota	Large (I64)

ISSW – Jednoduchý přepínač celočíselných signálů

Symbol bloku

Licence: [STANDARD](#)



Popis funkce

Blok ISSW je jednoduchý přepínač celočíselných vstupních signálů *i1* a *i2* na základě logického vstupu *SW*. Jestliže *SW* je *off*, pak výstup *n* je roven signálu *i1*. Jestliže *SW* je *on*, pak výstup *n* je roven signálu *i2*.

Tento blok propaguje kvalitu signálu. Více informací je uvedeno v sekci [1.4](#).

Vstup

<i>i1</i>	První celočíselný vstup bloku	Long (I32)
<i>i2</i>	Druhý celočíselný vstup bloku	Long (I32)
<i>SW</i>	Přepínací signál	Bool
	<i>off</i> ... je zvolen vstupní signál <i>i1</i>	
	<i>on</i> je zvolen vstupní signál <i>i2</i>	

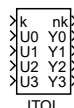
Výstup

<i>n</i>	Celočíselný výstupní signál	Long (I32)

ITOI – Transformace celých a binárních čísel

Symbol bloku

Licence: [STANDARD](#)



Popis funkce

Blok ITOI přiřazuje vstupnímu číslu k respektive binárnímu číslu $(U3\ U2\ U1\ U0)_2$ z množiny $\{0, 1, 2, \dots, 15\}$ výstupní číslo nk a jeho binární reprezentaci $(Y3\ Y2\ Y1\ Y0)_2$ z téže množiny. Příslušné zobrazení je popsáno tabulkou

k	0	1	2	...	15
nk	$n0$	$n1$	$n2$	...	$n15$

kde $n0, \dots, n15$ jsou dány převodním vektorem $fktab$. Je-li $BINF = off$, potom se za významný považuje vstup k , zatímco pro $BINF = on$ se za vstup bloku považuje číslo $(U3\ U2\ U1\ U0)_2$.

Tento blok propaguje kvalitu signálu. Více informací je uvedeno v sekci [1.4](#).

Vstup

k	Celočíselný vstupní signál	Long (I32)
$U0 \dots U3$	Binární vstup (maska)	Bool

Parametr

BINF	Výběr pomocí binárních vstupů off ... zakázáno (výběr přes iSW) on povoleno (výběr přes SW1 a SW2)	$\odot on$ Bool
fktab	Mapovací tabulka $\odot [0\ 1\ 2\ 3\ 4\ 5\ 6\ 7\ 8\ 9\ 10\ 11\ 12\ 13\ 14\ 15]$	Byte (U8)

Výstup

nk	Celočíselný výstupní signál	Long (I32)
$Y0 \dots Y3$	Binární výstup (maska)	Bool

NOT – Logická negace

Symbol bloku

Licence: [STANDARD](#)



Popis funkce

Blok NOT dělá logickou negaci vstupního signálu $Y = \neg U$.

Tento blok propaguje kvalitu signálu. Více informací je uvedeno v sekci [1.4](#).

Vstup

U	Logický vstupní signál	Bool

Výstup

Y	Logický výstupní signál	Bool

OR – Logický součet dvou signálů

Symbol bloku

Licence: [STANDARD](#)



Popis funkce

Blok **OR** dělá logický součet dvou vstupních signálů $Y = U1 \vee U2$. Pokud potřebujete pracovat s více vstupními signály, použijte jeden z bloků [ORQUAD](#), [OROCT](#) nebo [ORHEXD](#).

Tento blok propaguje kvalitu signálu. Více informací je uvedeno v sekci [1.4](#).

Vstup

U1	První logický vstup bloku	Bool
U2	Druhý logický vstup bloku	Bool

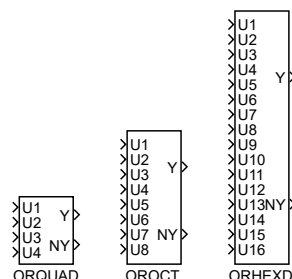
Výstup

Y	Výstup bloku, logický součet	Bool
NY	Negace výstupního signálu Y	Bool

ORQUAD, OROCT, ORHEXD – Logický součet více signálů

Symboly bloků

Licence: [STANDARD](#)



Popis funkce

Bloky **ORQUAD**, **OROCT**, **ORHEXD** provádí logický součet až šestnácti vstupních signálů **U1** až **U16**. Signály, jejichž seznam je uveden v parametru **n1** se před provedením logického součinu negují.

Pokud je tedy parametr **n1** prázdný, tak se provádí logický součet $Y = U1 \vee U2 \vee U3 \vee U4 \vee U5 \vee \dots \vee U16$. Pokud bude například **n1=1, 3..5**, pak $Y = \neg U1 \vee U2 \vee \neg U3 \vee \neg U4 \vee \neg U5 \vee U6 \vee \dots \vee U16$.

Pokud pracujete pouze se dvěma vstupními signály, zvažte použití bloku [OR](#).

Tento blok propaguje kvalitu signálu. Více informací je uvedeno v sekci [1.4](#).

Vstup

U1..U16	Logický vstupní signál	Bool

Parametr

n1	Seznam negovaných signálů	Long (I32)

Výstup

Y	Výstup bloku, logický součet	Bool
NY	Negace výstupního signálu Y	Bool

RS – Klopný obvod

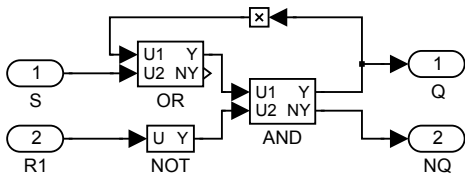
Symbol bloku

Licence: [STANDARD](#)



Popis funkce

Blok RS je klopný obvod, který v případě, že vstup S má hodnotu on, nastaví trvale výstup Q na on. Druhý vstupní signál R1 resetuje výstup Q na hodnotu off a to i tehdy, když vstup S má hodnotu on. Výstup NQ je pouhou negací výstupu Q.
Funkce bloku je dobře patrná z obrázku vnitřní struktury bloku.



Tento blok propaguje kvalitu signálu. Více informací je uvedeno v sekci [1.4](#).

Vstup

S	Nahození klopného obvodu	Bool
R1	Přednostní shození klopného obvodu	Bool

Výstup

Q	Stav klopného obvodu	Bool
NQ	Negace výstupního signálu Q	Bool

SR – Klopný obvod

Symbol bloku

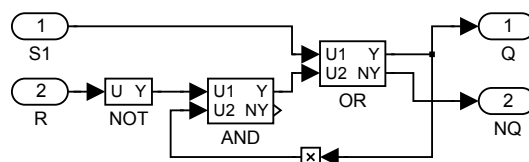
Licence: [STANDARD](#)



Popis funkce

Blok **SR** je klopný obvod, který v případě, že vstup **S1** má hodnotu **on**, nastaví trvale výstup **Q** na **on**. Druhý vstupní signál **R** resetuje výstup **Q** na hodnotu **off**, ale pouze tehdy, když vstup **S1** má hodnotu **off**. Výstup **NQ** je pouhou negací výstupu **Q**.

Funkce bloku je dobře patrná z obrázku vnitřní struktury bloku.



Tento blok propaguje kvalitu signálu. Více informací je uvedeno v sekci [1.4](#).

Vstup

S1	Přednostní nahození klopného obvodu	Bool
R	Shození klopného obvodu	Bool

Výstup

Q	Stav klopného obvodu	Bool
NQ	Negace výstupního signálu Q	Bool

TIMER – Vícefunkční časovač

Symbol bloku

Licence: [STANDARD](#)

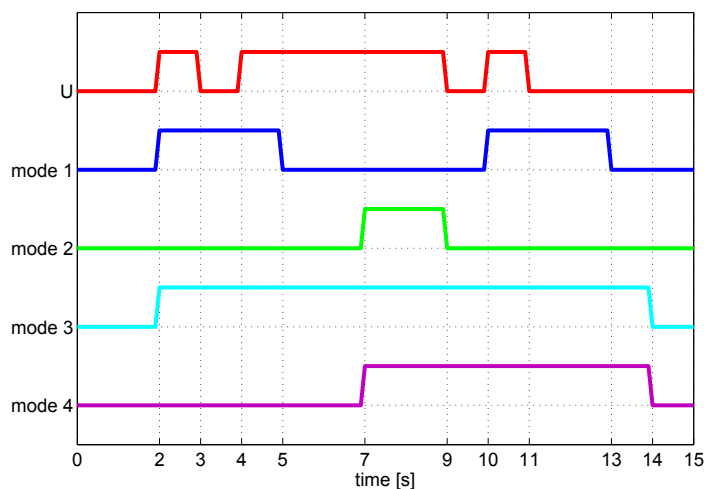


Popis funkce

Blok **TIMER** umožňuje buď vygenerovat impuls zadané délky **pt** (v sekundách) nebo filtrovat pulzy na vstupním signálu **U** užší než **pt** sekund. Režim funkce bloku se volí pomocí parametru **mode**. Podporované režimy jsou:

- **1: Pulse:** Na výstupu je pulz délky **pt** sekund, který začíná náběžnou hranou na vstupu **U**; další náběžné hrany na vstupu **U** během trvání pulzu jsou ignorovány.
- **2: Delayed ON:** Signál ze vstupu **U** je kopírován na výstup **Q** tak, že začátek impulsu na výstupu je opožděn o **pt** sekund proti začátku pulzu na vstupu; pulzy kratší než **pt** sekund se na výstupu neobjeví.
- **3: Delayed OFF:** Signál ze vstupu **U** je kopírován na výstup **Q** tak, že konec impulsu na výstupu je opožděn o **pt** sekund proti konci pulzu na vstupu; pokud je mezera mezi vstupními pulzy kratší než **pt** sekund, výstup je trvale aktivní.
- **4: Delayed change:** Výstupní signál **Q** se přepne na hodnotu vstupu **U** až tehdy, když je vstup po dobu **pt** sekund neměnný.

Následující obrázek ilustruje chování bloku v jednotlivých režimech při nastavení **pt = 3**:



Čítání času je možno pozastavit pomocí vstupu **HLD**. Vstup **R1** resetuje časovač. Signál pro reset má přednost před vstupem **U**, obdobně jako u bloku [RS](#).

Tento blok propaguje kvalitu signálu. Více informací je uvedeno v sekci [1.4](#).

Vstup

U	Signál spouštějící časovač	Bool
HLD	Pozastavení časovače	Bool
R1	Reset časovače	Bool

Parametr

mode	Režim činnosti časovače	⊙1 Long (I32)
	1 generovaný pulz	
	2 zpožděné zapnutí	
	3 zpožděné vypnutí	
	4 zpožděná změna	
pt	Doba časování [s]	⊙1.0 Double (F64)

Výstup

Q	Výstupní signál časovače	Bool
et	Doba uplynulá od startu časovače [s]	Double (F64)
rt	Zbývající doba [s]	Double (F64)

Kapitola 9

TIME – Bloky pro práci s časem

Obsah

DATE – Aktuální datum	328
DATETIME – Čtení, nastavování a konverze času	329
TC – Řízení časovače	332
TIME – Aktuální čas	334
TS – Aktuální časová značka	335
TS2NS – Rozdíl časových značek v nanosekundách	337
WSCH – Týdenní programátor	338

Knihovna TIME se specializuje na časově založené operace a plánování v systému REXYGEN. Obsahuje bloky jako [DATE](#), [TIME](#) a [DATETIME](#) pro manipulaci s datem a poskytuje základní nástroje pro práci s časovými daty. Knihovna obsahuje [TC](#) pro ovládání vnitřního časovače. Kromě toho se pro plánování používá [WSCH](#), což umožňuje efektivní správu časově závislých úloh. Tato knihovna je zvláště cenná pro systémy vyžadující přesné řízení času a možnosti plánování.

DATE – Aktuální datum

Symbol bloku

Licence: [STANDARD](#)



Popis funkce

Výstupy bloku **DATE** odpovídají datu operačního systému. Pro pokročilé operace s časem a datem použijte blok [DATETIME](#). Prvním dnem v týdnu je neděle (číslováno od 1).

Tento blok nepropaguje kvalitu signálu. Více informací je uvedeno v sekci [1.4](#).

Parametr

tz	Časové pásmo	⊙1	Long (I32)
	1 lokální čas		
	2 UTC		

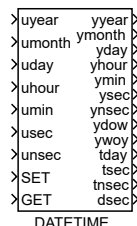
Výstup

year	Rok	Long (I32)
month	Měsíc	Long (I32)
day	Den	Long (I32)
dow	Den v týdnu	Long (I32)

DATETIME – Čtení, nastavování a konverze času

Symbol bloku

Licence: [STANDARD](#)



Popis funkce

Blok **DATETIME** je určen pro pokročilé operace s časem řídicího systému **REXYGEN** a operačního systému.

Blok umožňuje synchronizaci hodin operačního systému a řídicího systému **REXYGEN**. V okamžiku spuštění exekutivy systému **REXYGEN** jsou hodiny synchronizovány, ale během dlouhodobého provozu se mohou tyto dva údaje rozcházet (např. při přechodu na letní čas). Pokud je potřeba provést opětovnou synchronizaci, hodiny systému **REXYGEN** se při náběžné hraně (**off**→**on**) na vstupu **SET** aktualizují dle vstupů a parametrů bloku.

Je však důrazně doporučeno neaktualizovat hodiny systému **REXYGEN**, pokud je řízený stroj či technologie v provozu, neboť by to mohlo vést k nepředvídatelnému chování.

Pokud je potřeba číst nebo konvertovat údaje o čase, je možno příslušnou akci spustit náběžnou hranou (**off**→**on**) na vstupu **GET** a hodnoty přečíst na výstupech bloku. Výstupy začínající na 't' označují celkový počet daných jednotek od 1.1.2000 UTC.

Pokud jsou nastaveny parametry **getper** a **setper** na nenulové hodnoty, je čtení a nastavování hodin prováděno periodicky.

Při menší odchylce hodin systému **REXYGEN** a operačního systému, než udává parametr **settol**, nejsou hodiny systému **REXYGEN** nastaveny jednorázově, synchronizace probíhá postupně. Toho je dosaženo zanedbatelnými změnami v časování exekutivy systému **REXYGEN**, čímž po nějaké době dojde k dosažení synchronizace. Následně je použito standardní časování systému **REXYGEN**.

Pro jednoduché čtení data a/nebo času použijte bloky [DATE](#) a [TIME](#).

Tento blok nepropaguje kvalitu signálu. Více informací je uvedeno v sekci [1.4](#).

Vstup

uyear	Vstup pro nastavení roku	Long (I32)
umonth	Vstup pro nastavení měsíce	Long (I32)
uday	Vstup pro nastavení dne	Long (I32)
uhour	Vstup pro nastavení hodin	Long (I32)

umin	Vstup pro nastavení minut	Long (I32)
usec	Vstup pro nastavení sekund	Long (I32)
unsec	Vstup pro nastavení nanosekund ↓-9.22E+18 ↑9.22E+18	Large (I64)
SET	Nastavení času pomocí náběžné hrany	Bool
GET	Přečtení času pomocí náběžné hrany	Bool

Parametr

isetmode	Zdroj podle kterého nastavit čas	⊙1	Long (I32)
	1 čas OS		
	2 vstupy		
	3 unsec		
	4 usec		
	5 unsec relativně		
	6 usec -jen RTC čas		
igetmode	Zdroj ze kterého přečíst čas	⊙6	Long (I32)
	1 čas OS		
	2 vstupy		
	3 unsec		
	4 usec		
	5 uday		
	6 čas systému REXYGEN		
	7 čas RTC		
settol	Tolerance pro nastavení hodin systému REXYGEN [s]	⊙1.0	Double (F64)
setper	Perioda nastavování času [s] (0=bez opakování)		Double (F64)
getper	Perioda čtení času [s] (0=bez opakování)	⊙0.001	Double (F64)
FDOW	První den v týdnu je neděle		Bool
	off ... týden začíná pondělím		
	on ... týden začíná nedělí		
tz	Časové pásmo	⊙1	Long (I32)
	1 lokální čas		
	2 UTC		

Výstup

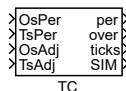
yyear	Rok	Long (I32)
ymonth	Měsíc	Long (I32)
yday	Den	Long (I32)
yhour	Hodiny	Long (I32)
ymin	Minuty	Long (I32)
ysec	Sekundy	Long (I32)
ynsec	Nanosekundy	Long (I32)
ydw	Den v týdnu	Long (I32)
ywoy	Týden v roce	Long (I32)
tday	Počet dní od začátku epochy	Long (I32)
tsec	Počet sekund od začátku epochy	Long (I32)

tnsec	Počet nanosekund od začátku epochy	Large (I64)
dsec	Počet sekund od půlnoci	Long (I32)

TC – Řízení časovače

Symbol bloku

Licence: [STANDARD](#)



Popis funkce

Blok TC řídí interní časovač systému REXYGEN. Bloku umožňuje modifikovat základní tik algoritmu (která se zadává parametrem `tick` bloku [EXEC](#)) a to jak skutečnou délku tiků, tak i logickou délku (kolik sekund se přičte do časové značky při každém tik). Parametr `EXEC:tick` nastavuje logickou i fyzickou periodu a také periodu spouštění bloků (některé bloky potřebují pro diskretizaci algoritmu). Perioda pro bloky není blokem TC ovlivněna.

Skutečná (fyzická) perioda se nastavuje vstupem `OsPer`. Dále je možné nastavit posun tiků o několik sekund. To se provede nastavením posunu na vstup `OsAdj` na jeden tik. Aby nebylo příliš narušeno časování, jsou větší posuny realizovány tak, že je dočasně snížena nebo zvýšena perioda tiků, dokud nedojde k požadovanému posunu. Jak se změní perioda ovlivňuje parametr `OsMax`.

Příklad: předpokládejme periodu tiků 0.1s a `OsMax=0.2`, pak nastavení `OsAdj=1.0` (na jeden tik) dočasně zvýší periodu na 0.12s (tj. o 20% jak určuje parametr `OsMax`), dokud nedojde k celkovému posunu 1sd, tj. na 50 tiků.

Logická perioda se řídí stejně s využitím vstupů/parametrů `TsPer`, `TsAdj`, `TsMax`.

Poznámka 1: Nepřipojené vstupy a vstupy s hodnotou 0 jsou ignorovány (nevyvolávají žádnou akci).

Poznámka 2: Nastavení skutečné periody není na windows platformách momentálně podporováno.

Poznámka 3: Hlavní účel bloku je synchronizovat čas/tiky systému REXYGEN s jiným systémem, takže změny period i offsety se předpokládají malé. Pro simulační a ladící účely je možné změnit periodu výrazně a tím zrychlit pomalu probíhající proces (nebo naopak zpomalit příliš rychle probíhající proces). To je potřeba dělat s rozmyslem, protože návaznost na další systémy pomocí driverů v podstatě přestane fungovat a také se musí při zkrácené periodě stihnout všechny výpočty. Navíc se v tomto případě objevují v logu warningy o chybějících ticích, špatné periodě a pod. Pro tyto účely je lépe použít simulační režim.

Tento blok nepropaguje kvalitu signálu. Více informací je uvedeno v sekci [1.4](#).

Vstup

<code>OsPer</code>	Skutečná perioda tiků [s]	Double (F64)
<code>TsPer</code>	Perioda tiků pro časovou značku[s]	Double (F64)

OsAdj	Posun skutečné polohy tiků [s]	Double (F64)
TsAdj	Posun časové značky tiků [s]	Double (F64)

Parametr

OsMax	Maximální relativní změna tiků pro vstup OsAdj ↓0.0 ↑1.0 ⊙0.1	Double (F64)
TsMax	Maximální relativní změna tiků pro vstup TsAdj ↓0.0 ↑1.0 ⊙0.1	Double (F64)

Výstup

per	Skutečná délka posledního tiků [s]	Double (F64)
over	Počet ztracených tiků v poslední periodě	Long (I32)
ticks	Počet tiků od startu	Large (I64)
SIM	Časovač v simulačním režimu	Double (F64)

TIME – Aktuální čas

Symbol bloku

Licence: [STANDARD](#)



Popis funkce

Výstupy bloku **TIME** odpovídají času operačního systému. Pro pokročilé operace s časem a datem použijte blok [DATETIME](#).

Tento blok nepropaguje kvalitu signálu. Více informací je uvedeno v sekci [1.4](#).

Parametr

tz	Časové pásmo	⊙1	Long (I32)
	1 lokální čas		
	2 UTC		

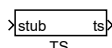
Výstup

hour	Hodiny	Long (I32)
min	Minuty	Long (I32)
sec	Sekundy	Long (I32)

TS – Aktuální časová značka

Symbol bloku

Licence: [STANDARD](#)



Popis funkce

Blok **TS** generuje na výstupu **ts** časovou značku ze zdroje určeného parametrem **source**. Vstup **stub** slouží pouze k zajištění správného pořadí vykonávání bloku v programu.

Parametrem **source** lze přepínat mezi několika různými zdroji času:

- 1: **CORETIMER** - je primární časová značka využívaná v systému REXYGEN a vyjadřuje počet nanosekund od 1. 1. 2000.
- 2: **CORETIMER (přesný)** - funguje podobně jako **CORETIMER**, ale časová značka je aktualizována v okamžiku vykonávání bloku, k čemuž se využívá jiný zdroj, obvykle **RTC** nebo **PFC**.
- 3: **RTC (UTC)** - vrací časovou značku ve formátu koordinovaného světového času.
- 4: **RTC (lokální čas)** - vrací časovou značku ve formátu lokálního času (bere v ohledu časová pásma).
- 5: **PFC** - vychází z funkce **QueryPerformanceCounter** na systémech Windows a vrací časovou značku systému s rozlišením 1 ns.
- 6: **TSC** - je nejrychlejší zdroj časové značky, využívajících na x86 procesorech instrukci **RDTSC** a na ARM64 procesorech instrukci **CNTVCT_EL0**. Na ostatních platformách je časová značka **TSC** shodná s časovou značkou **PFC**.

Tento blok nepropaguje kvalitu signálu. Více informací je uvedeno v sekci [1.4](#).

Vstup

stub	Falešný vstup pro řazení bloku	Any

Parametr

source	Zdroj časové značky	⊙1 Word (U16)
1	 CORETIMER	
2	 CORETIMER (přesný)	
3	 RTC (UTC)	
4	 RTC (lokální čas)	
5	 PFC	
6	 TSC	

Výstup

ts	Časová značka	Large (I64)

TS2NS – Rozdíl časových značek v nanosekundách

Symbol bloku

Licence: [STANDARD](#)



Popis funkce

Blok **TS2NS** generuje na výstupu **ns** rozdíl časových značek přivedených na vstupy **start** a **end** (například z bloků [TS](#)). Při počítání rozdílu musí mít všechny bloky **TS** a **TS2NS** stejný zdroj v parametru **source**, jinak je výsledek nesmyslný. Blok poskytuje na výstupu **freq** frekvenci časové značky. Rozdíl časových značek je v nanosekundách (ns). Pro podrobnější informace o zdrojích časových značek viz blok [TS](#).

Tento blok nepropaguje kvalitu signálu. Více informací je uvedeno v sekci [1.4](#).

Vstup

start	Počáteční časová značka	↓-9.22E+18 ↑9.22E+18	Large (I64)
end	Konečná časová značka	↓-9.22E+18 ↑9.22E+18	Large (I64)

Parametr

source	Zdroj časové značky	⊙1 Word (U16)
1	 CORETIMER	
2	 CORETIMER (přesný)	
3	 RTC (UTC)	
4	 RTC (lokální čas)	
5	 PFC	
6	 TSC	

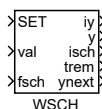
Výstup

ns	Časový úsek v nanosekundách	Large (I64)
freq	Frekvence časové značky	Large (I64)

WSCH – Týdenní programátor

Symbol bloku

Licence: [STANDARD](#)



Popis funkce

Blok **WSCH** je určen pro generování týdenních programů, například pro vytápění (den, noc, útlum), větrání (high, low, off), osvětlení, zavlažování apod. Jeho výstupy mohou být využity pro spínání jednotlivých zařízení nebo pro regulaci jejich výkonu.

V běžném provozu jsou v průběhu týdne na výstupech **iy** a **y** generovány hodnoty dle tabulky **wst**, která obsahuje trojice hodnot *den-hodina-hodnota*. Například zápis [2 6.5 21.5] znamená, že se v úterý v 6:30 hodin ráno nastaví na výstup **y** hodnota 21.5 a na výstupu **iy** bude hodnota 22 (zaokrouhlení na celé číslo). Jednotlivé trojice hodnot se oddělují středníkem.

Dny jsou číslovány od 1 (pondělí) do 7 (neděle). Vyšší čísla je možno využít pro speciální denní programy, které je možno vynutit pomocí vstupu **fsch** nebo tabulky speciálních dnů **specdays**. Aktuálně platný denní program je indikován výstupem **isch**.

Rovněž je možné dočasně nastavit výstupní hodnotu pomocí vstupů **SET** a **val**. Při náběžné hraně na vstupu **SET** (off→on) je hodnota **val** zkopírována na výstup **y** a výstup **isch** je přenastaven na hodnotu 0. Ruční hodnota zůstává nastavena, dokud:

- nenastane další přepnutí výstupní hodnoty dle tabulky **wst** nebo
- není přenastavena pomocí další náběžné hrany na vstupu **SET** nebo
- není vynucen jiný denní program pomocí vstupu **fsch**.

Seznam speciálních dní **specdays** lze využít pro vynucení konkrétního denního programu v daný den. Například ve dnech státních svátků můžeme vynutit nedělní režim. Datum se zadává ve formátu **YYYYMMDD**. Zápis [20160328 7] tak znamená, že 28. března 2016 se má generovat nedělní program. Jednotlivé dvojice hodnot se oddělují středníkem.

Výstupy **trem** a **ynext** mohou být využity, pokud je potřeba provést nějaké úkony v předstihu ještě před přepnutím výstupních hodnot **iy** a **y**.

Výstup **iy** je určen pro přímé napojení na funkční bloky se vstupy typu Boolean (konverze typu **long** na **bool** se provádí automaticky).

Parametr **nmax** určuje, kolik paměti je alokováno pro pole **wst** a **codespecdays**. Při **nmax** = 100 může parametr **wst** obsahovat až 100 trojic *den-hodina-hodnota*. Pro běžné použití není potřeba velikost **nmax** měnit.

Tento blok nepropaguje kvalitu signálu. Více informací je uvedeno v sekci 1.4.

Vstup

SET	Nastavení výstupu pomocí náběžné hrany	Bool
val	Hodnota pro dočasné nastavení výstupu	Double (F64)
fsch	Vynucený denní program (0=provoz dle týdenního programu)	Long (I32)
	0 Standardní týdenní program	
	1 Pondělí	
	2 Úterý	
	3 Středa	
	4 Čtvrtek	
	5 Pátek	
	6 Sobota	
	7 Neděle	
	8 a více Další denní program definovaný tabulkou wst	

Parametr

tz	Časové pásmo	⊙1	Long (I32)
	1 lokální čas		
	2 UTC		
nmax	Velikost alokovaných polí	↓10 ↑1000000 ⊙100	Long (I32)
imode	Rezervováno pro interní použití		Long (I32)
user	Rezervováno pro speciální editor		String
wst	Tabulka týdenního programu (den-hodina-hodnota)		Double (F64)
	⊙[1 0.01 18.0; 2 6.0 22.0; 2 18.0 18.0; 3 6.0 22.0; 3 18.0 18.0; 4 6.0 22.0; 4 18.0 18.0		
specdays	Seznam speciálních dní (datum-denní program)		Long (I32)
	⊙[20150406 1; 20151224 1; 20151225 1; 20151226 1; 20160101 1; 20160328 1; 20170417 1; 20		

Výstup

iy	Celočíselná výstupní hodnota	Long (I32)
y	Výstupní hodnota	Double (F64)
isch	Identifikace denního programu	Long (I32)
trem	Čas zbývajících v aktuálním intervalu [s]	Double (F64)
ynext	Výstupní hodnota v dalším intervalu	Double (F64)

Kapitola 10

ARC – Archivace dat

Obsah

ACD – Archivní komprese s použitím delta kritéria	344
ACLEAR – Vynucené smazání archivu	346
AFLUSH – Vynucené zapsání archivu	347
ALB, ALBI – Alarmy pro logickou hodnotu	348
ALM, ALMI – Uložení hodnoty do alarmu	350
ALN, ALNI – Alarmy pro číselnou hodnotu	351
ARS – Uložení hodnoty do archivu	354
TRND – Záznam trendů v reálném čase	356
TRNDV – Záznam trendů v reálném čase (pro vektory)	359

Exekutiva systému REXYGEN se skládá z různých propojených podsystémů (podsystém v reálném čase, diagnostický podsystém, podsystém ovladačů atd.). Jeden z těchto podsystémů je *archivační podsystém*. Archivační podsystém se stará o zaznamenávání historie řídicího algoritmu.

Funkční bloky lze rozdělit do skupin podle jejich použití:

- Bloky pro generování alarmů a událostí – [ALB](#), [ALBI](#), [ALM](#), [ALMI](#), [ALN](#), [ALNI](#), [ARS](#).
- Bloky pro zaznamenávání trendů – [ACD](#), [TRND](#), [TRNDV](#).
- Bloky pro manipulaci s archivem – [AFLUSH](#), [ACLEAR](#).

Funkce archivačního subsystému

Archiv v systému REXYGEN uchovává historii událostí, alarmů a trendů vybraných signálů. V každém cílovém zařízení může být až 15 archivů. Typy archivů jsou uvedeny níže:

Archiv v paměti RAM – Vhodný pro krátkodobé ukládání dat. Výhodou je rychlý přístup k uloženým datům, nevýhodou ztráta dat po restartu systému.

Archiv v zálohované paměti. – Podobný archivu v paměti RAM. Největší výhodou je zachovávání uložených dat i při opakovaných restartech systému, navíc přístup k datům zůstává velmi rychlý. Nevýhodou může být někdy jeho nepříliš velká kapacita (závisí na konkrétní hardwarové platformě).

Archiv v souboru na disku. – Archivy na disku jsou soubory speciálního formátu. Výhodami jsou snadná přenositelnost (kopírování) a zejména velký rozsah dat omezený jen kapacitou disku. Nevýhodou je pomalejší přístup k datům.

Daná hardwarová platforma nemusí podporovat všechny druhy archivů. Příznaky podporovaných druhů archivů jsou součástí verze řídicího systému cílového zařízení a lze je zjistit v diagnostickém panelu v REXYGEN Studio po kliknutí na jméno cílového zařízení (IP adresu) ve stromu exekutivy. Nachází se na kartě **Target** v levé spodní části.

Obecné vlastnosti archivů

Archivační a trendovací bloky mají několik společných vlastností, které jsou uvedeny níže.

Seznam archivů

Seznam archivů je v blocích specifikován parametrem **arc** ve formě např. 1,3..5,8. Podrobnosti o číslování archivů jsou v bloku **ARC**. Programy třetích stran (Simulink, OPC klienti atd.) pracují s celým číslem, které je bitová maska – pro uvedený příklad je to 157, v binární podobě 10011101.

Kód identifikace události

Kód identifikace události v archivu **id** musí být unikátní v celém cílovém zařízení s řídicím systémem REXYGEN (tj. ve všech archivačních blocích). Pokud je **id** = 0, alarm se negeneruje. Pro **id** = -1 se alarm identifikuje jménem (tj. název bloku musí být stejný jako sloupec **Name** v definiční tabulce alarmů).

Typy událostí a alarmů

Události a alarmy jsou v systému REXYGEN rozlišeny pomocí parametru **lv1**. Pokud je $1 \leq lv1 \leq 127$, jedná se o alarm, u něhož se do archivu ukládá jeho začátek, konec i potvrzení. Rozsah $128 \leq lv1 \leq 255$ je určen pro události, u nichž se zapisuje pouze okamžik, kdy daná událost nastala.

Nastavení jednotlivých úrovní si může každý zvolit podle svého uvážení. Níže je pro ilustraci uvedeno nastavení v systémovém logu, kde mají čísla na pozici jednotek a desítek následující význam:

- **x1**: Zprávy operačního systému
- **x2**: Zprávy jádra
- **x3**: Diagnostické zprávy
- **x4**: Zprávy z bloků

- **x5**: Zprávy z archivu
- **x6**: Zprávy I/O ovladačů
- **1x**: Zprávy zápisu I/O ovladačů
- **2x**: Zprávy čtení I/O ovladačů
- **3x**: Verbose zprávy
- **4x**: Informační zprávy
- **5x**: Varovné zprávy
- **6x**: Chybové zprávy

ACD – Archivní komprese s použitím delta kritéria

Symbol bloku

Licence: [STANDARD](#)



Popis funkce

Blok ACD (Archive Compression using Delta criterion) je určen pro ukládání komprimovaných analogových signálů do archivu pomocí archivních událostí.

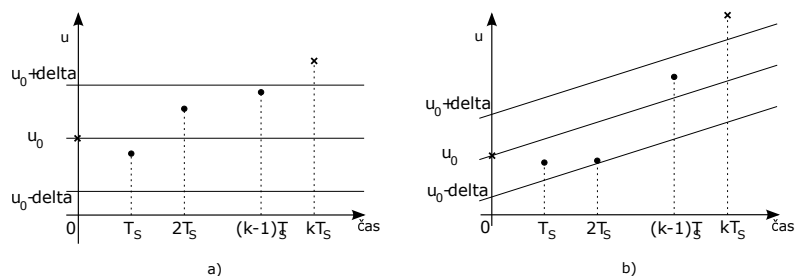
Základní myšlenkou bloku je archivovat vstupní signál u jen tehdy, pokud se mění. Doba mezi uložením dvou po sobě následujících hodnot signálu je v intervalu $\langle t_{\min}, t_{\max} \rangle$ sekund (doby jsou zaokrouhleny na nejbližší násobek periody vzorkování). Pokud se hodnota signálu „hodně“ mění, ukládá se signál jednou za čas $t_{\min}$, pokud se hodnota signálu mění „málo“ nebo je konstantní, ukládá se signál jednou za čas $t_{\max}$. Po spuštění bloku se vždy uloží první hodnota vstupu u , označme ji u_0 . Přesná pravidla ukládání dalších vzorků jsou určena vstupem delta a parametrem TR .

Seznam archivů pro ukládání je specifikován parametrem arc , např. 1, 3...5, 8. Událost bude uložena ve všech specifikovaných archivech. Každý archivační blok musí mít jednoznačný kód identifikace události v archivu daný parametrem id . Pro více informací o těchto parametrech viz úvod kapitoly 10.

Je-li $TR=\text{off}$, testuje se podmínka $|u - u_0| > \text{delta}$. Pokud je splněna a od minulého uložení uplynul alespoň čas $t_{\min}$ uloží se tato hodnota u do archivu a nastaví se $u_0 = u$. Je-li podmínka splněna dříve než za čas $t_{\min}$ od posledního uložení nastaví se chybový výstup E na 1 a počká se s uložením na první vzorek po uplynutí času $t_{\min}$, v tomto okamžiku se nastavuje $E=0$. Pak se celý postup opakuje od začátku.

Je-li $TR=\text{on}$, pracuje blok tak, že ukládá první vzorek, který se odchyluje o více než toleranci delta od signálu s kompenzovaným trendem. Podmínka na minimální čas ukládání platí obdobně jako v předcházejícím případě.

Chování bloku v obou případech ukazuje následující obrázek: a) pro $TR=\text{off}$, b) pro $TR=\text{on}$. Ukládané vzorky jsou označeny symbolem $\times$.



Tento blok nepropaguje kvalitu signálu. Více informací je uvedeno v sekci 1.4.

Vstup

<code>u</code>	Komprimovaně ukládaný signál		Double (F64)
<code>delta</code>	Práh pro ukládání signálu do archivu	↓0.0 ↑1e+10	Double (F64)

Parametr

<code>acls</code>	Třída alarmu (typ proměnné)	⊙8	Byte (U8)
	1 Bool		
	2 Byte (U8)		
	3 Short (I16)		
	4 Long (I32)		
	5 Word (U16)		
	6 DWord (U32)		
	7 Float (F32)		
	8 Double (F64)		
	10 Large (I64)		
<code>arc</code>	Seznam archivů pro zápis alarmů		Word (U16)
<code>id</code>	Unikátní identifikátor události v archivu	⊙1	Word (U16)
<code>tmin</code>	Nejkratší čas mezi dvěma uloženími [s]	↓0.001 ↑1000000.0 ⊙1.0	Double (F64)
<code>tmax</code>	Nejdelší čas mezi dvěma uloženími [s]	↓1.0 ↑1000000.0 ⊙1000.0	Double (F64)
<code>TR</code>	Vyhodnocování trendu signálu	⊙on	Bool
	off ... zakázáno		
	on ... povoleno		
<code>Desc</code>	Bližší popis události	⊙Value Description	String

Výstup

<code>y</code>	Poslední hodnota uložená do archivu		Double (F64)
<code>E</code>	Příznak chyby		Bool
	off ... bez chyby		
	on ... nastala chyba		

ACLEAR – Vynucené smazání archivu

Symbol bloku

Licence: [STANDARD](#)



Popis funkce

Blok ACLEAR slouží k vymazání obsahu archivu v okamžiku náběžné hrany na vstupu CLEAR. Seznam archivů, které mají být vymazány, je specifikován parametrem **arc**, např. 1,3..5,8. Pro více informací viz Kapitola [10](#).

Tento blok nepropaguje kvalitu signálu. Více informací je uvedeno v sekci [1.4](#).

Vstup

CLEAR	Smazání archivů (náběžná hrana)	Bool

Parametr

arc	Seznam archivů pro zápis alarmů	Word (U16)

AFLUSH – Vynucené zapsání archivu

Symbol bloku

Licence: [STANDARD](#)



Popis funkce

Blok **AFLUSH** slouží k vynucenému zapsání dat archivu na disk v situaci, kdy hrozí vypnutí napájení řídicího systému, které by vedlo ke ztrátě archivních dat, která ještě nebyla uložena na disk. V okamžiku náběžné hrany na vstupu **FLUSH** (**off**→**on**) se uloží data na disk bez ohledu na nastavení parametru **period** bloku [ARC](#). Seznam archivů je specifikován parametrem **arc**, např. 1,3..5,8. Pro více informací viz Kapitola [10](#).

Tento blok nepropaguje kvalitu signálu. Více informací je uvedeno v sekci [1.4](#).

Vstup

FLUSH	Vynucené zapsání archivů	Bool

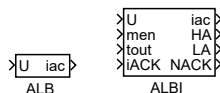
Parametr

arc	Seznam archivů pro zápis alarmů	Word (U16)

ALB, ALBI – Alarmy pro logickou hodnotu

Symbole bloků

Licence: [STANDARD](#)



Popis funkce

Bloky ALB a ALBI generují alarmy nebo události při změnách logického vstupního signálu *U*. Výstup *iac* indikuje aktuální kód alarmu (události). Parametr (nebo vstup) *men* volí, zda se indikuje náběžná hrana (*off*→*on*), což odpovídá hornímu alarmu (HA), sestupná hrana (*on*→*off*), což odpovídá dolnímu alarmu (LA), nebo obě hrany vstupního signálu.

Blok ALBI je rozšířením bloku ALB. Bloky se liší pouze tím, že vstupy bloku ALBI: *men*, *tout*, *iACK* jsou parametry bloku ALB. Blok ALB nemá výstupy HA, LA a NACK.

Události a alarmy jsou v systému REXYGEN rozlišeny pomocí parametru *lv1*. Seznam archivů pro zápis je specifikován parametrem *arc* ve formě např. 1,3..5,8. Každý archivační blok musí mít jednoznačný kód identifikace události v archivu daný parametrem *id*. Pro více informací o těchto parametrech viz úvod kapitoly 10. Kladné hodnoty kódů výstupu *iac* mohou být sčítány, např. hodnota 514 značí, že je nepotvrzený horní alarm. Ne všechny kombinace však mají smysl.

Poznámka 1: Vstup (parametr) *iACK* se automaticky nuluje po zpracování blokem. Potvrzení alarmu se předpokládá z vizualizace operátorem, tak aby nebylo nutné zapisovat ještě 0 dalším dotazem. Je to podobný princip jako parametr *BSTATE* v bloku [MP](#).

Poznámka 2: Do parametru *Desc* lze vkládat formátovací příkazy (hodnoty připojené k alarmu, vícejazyčný text). Jejich podrobný popis je uveden u bloku [ALARMS](#).

Tento blok nepropaguje kvalitu signálu. Více informací je uvedeno v sekci [1.4](#).

Vstup

<i>U</i>	Logický vstupní signál	Bool
<i>men</i>	Povolení alarmů (maska)	Long (I32)
	0 žádný alarm není povolen	
	1 povoleno generování dolního alarmu (LA)	
	2 povoleno generování horního alarmu (HA)	
	3 povoleno generování obou alarmů	
<i>tout</i>	Zpoždění aktivace alarmu [s]	↓0.0 Double (F64)

iACK	Potvrzení alarmu (maska)	Byte (U8)
1	 potvrzení dolního alarmu (LA)	
2	 potvrzení horního alarmu (HA)	
3	 potvrzení obou alarmů	

Parametr

arc	Seznam archivů pro zápis alarmů	Word (U16)
id	Unikátní identifikátor události v archivu	⊙1 Word (U16)
lvl	Úroveň (závažnost) alarmu	↓1 ⊙1 Byte (U8)
group	Skupina, do které alarm patří	↓0 ↑9.22337E+18 Large (I64)
Desc	Popis alarmu	⊙Alarm Description String

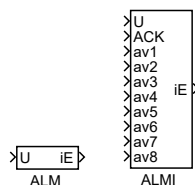
Výstup

iac	Kód aktuálního alarmu	Long (I32)
0	 signál v mezích	
1	 dolní alarm (LA) je aktivní	
2	 horní alarm (HA) je aktivní	
256	... dolní alarm (LA) není potvrzen (NACK)	
512	... horní alarm (HA) není potvrzen (NACK)	
HA	Indikátor horního alarmu	Bool
LA	Indikátor dolního alarmu	Bool
NACK	Indikátor nepotvrzeného alarmu	Bool

ALM, ALMI – Uložení hodnoty do alarmu

Symbole bloků

Licence: [STANDARD](#)



Popis funkce

Bloky ALM a ALMI slouží ke generování alarmů. Alarm je aktivován, pokud vstup U přejde do stavu **on**. Každý alarm musí být definován pomocí bloku [ALARMS](#) a je jednoznačně rozpoznán podle parametru *id* (viz [10](#)). Alarmy, které jsou aktivní (tj. s **U=on**), lze zobrazit ve vizualizační komponentě **AlarmsTable**. Záznamy o změně stavu alarmu a jeho potvrzení se ukládají do archivu, pokud je to povoleno v konfiguraci bloku [ALARMS](#). Alarm lze potvrdit aktivací parametru **ACK=on**.

Poznámka: Systém umožňuje zobrazení stavu potvrzení, potvrzení aktivního alarmu a může tento stav zobrazovat jak v HMI, tak v archivu. Avšak systém REXYGEN s potvrzením dále nijak neoperuje a žádná funkcionality na něm nezávisí. Rozhodnutí o potvrzování alarmů závisí na nastavení filtru ve vizualizaci a na konkrétním návrhu systému.

Tento blok nepropaguje kvalitu signálu. Více informací je uvedeno v sekci [1.4](#).

Vstup

U	Logický vstupní signál	Bool
ACK	Potvrzení alarmu (maska)	Bool
av1..av8	Hodnota spojená s alarmem	↓-1.79769e+308 ⊙ -1.79769e+308 Double (F64)

Parametr

<i>id</i>	Unikátní identifikátor události v archivu	↓-1 ↑65535 ⊙ -1 Long (I32)

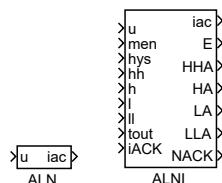
Výstup

iE	Kód chyby	Error

ALN, ALNI – Alarmy pro číselnou hodnotu

Symbole bloků

Licence: [STANDARD](#)



Popis funkce

Bloky ALN a ALNI jsou určeny pro generování dvouúrovňových alarmů nebo událostí při překročení (podkročení) číselné hodnoty vstupu **u** některou z alarmových mezí. Výstup **iac** indikuje stav alarmu (události). Parametr (nebo vstup) **men** určuje, které z mezí jsou hlídány. Lze volit mezi těmito mezemi:

- **LA** - dolní alarm
- **HA** - horní alarm
- **LLA** - druhý dolní alarm
- **HHA** - druhý horní alarm

a jejich kombinacemi.

Blok ALNI je rozšířením bloku ALN. Bloky se liší pouze tím, že většina vstupů bloku ALNI jsou u bloku ALN parametry. Blok ALN nemá výstupy HHA, HA, LA, LLA a NACK.

Jednotlivé hodnoty mezí lze nastavit parametry (vstupy) **l**, **h**, **ll** a **hh**. Hodnota **hys** určuje hysterezi alarmu. Výstupy HHA, HA, LA, LLA a NACK indikují, zda je daný alarm aktivní/nepotvrzený.

Události a alarmy jsou v systému REXYGEN rozlišeny pomocí parametru **lvl**. Seznam archivů pro zápis je specifikován parametrem **arc**, např. **1,3..5,8**. Kód identifikace události v archivu **id** musí být unikátní v celém cílovém zařízení s řídicím systémem REXYGEN. Pro více informací o těchto parametrech viz úvod kapitoly [10](#). Kladné hodnoty kódů výstupu **iac** mohou být sčítány, např. hodnota **514** značí, že je nepotvrzený horní alarm. Ne všechny kombinace však mají smysl.

Poznámka 1: Vstup (parametr) **iACK** se automaticky nuluje po zpracování blokem. Potvrzení alarmu se předpokládá z vizualizace operátorem, tak aby nebylo nutné zapisovat ještě 0 dalším dotazem. Je to podobný princip jako parametr **BSTATE** v bloku [MP](#).

Poznámka 2: Do parametru **Desc** lze vkládat formátovací příkazy (hodnoty připojené k alarmu, vícejazyčný text). Jejich podrobný popis je uveden u bloku [ALARMS](#).

Tento blok nepropaguje kvalitu signálu. Více informací je uvedeno v sekci 1.4.

Vstup

u	Analogový vstupní signál		Double (F64)
men	Povolení alarmů (maska)	⊙15	Long (I32)
	0 žádný alarm není povolen		
	1 povoleno generování dolního alarmu (LA)		
	2 povoleno generování horního alarmu (HA)		
	3 povoleno generování LA a HA		
	4 povoleno generování druhého dolního alarmu (LLA)		
	5 povoleno generování LA a LLA		
	6 povoleno generování HA a LLA		
	7 povoleno generování LA, HA a LLA		
	8 povoleno generování druhého horního alarmu (HHA)		
	9 povoleno generování LA a HHA		
	10 povoleno generování HA a HHA		
	11 povoleno generování LA, HA a HHA		
	12 povoleno generování LLA a HHA		
	13 povoleno generování LA, LLA a HHA		
	14 povoleno generování HA, LLA a HHA		
	15 povoleno generování všech alarmů		
hys	Hystereze alarmu	↓1e-10 ↑1e+10	Double (F64)
hh	Mez pro druhý horní alarm		Double (F64)
h	Mez pro horní alarm		Double (F64)
l	Mez pro dolní alarm		Double (F64)
ll	Mez pro druhý dolní alarm		Double (F64)
tout	Zpoždění aktivace alarmu [s]	↓0.0	Double (F64)
iACK	Potvrzení alarmu (maska)		Byte (U8)
	1 potvrzení dolního alarmu (LA)		
	2 potvrzení horního alarmu (HA)		
	4 potvrzení druhého dolního alarmu (LLA)		
	8 potvrzení druhého horního alarmu (HHA)		

Parametr

acls	Třída alarmu (typ proměnné)	⊙8	Byte (U8)
	1 Bool		
	2 Byte (U8)		
	3 Short (I16)		
	4 Long (I32)		
	5 Word (U16)		
	6 DWord (U32)		
	7 Float (F32)		
	8 Double (F64)		
	10 Large (I64)		
arc	Seznam archivů pro zápis alarmů		Word (U16)
id	Unikátní identifikátor události v archivu	⊙1	Word (U16)

lvl1	Úroveň (závažnost) horního a dolního alarmu	↓1 ⊙1	Byte (U8)
lvl2	Úroveň (závažnost) druhých horních a dolních alarmů	↓1 ⊙10	Byte (U8)
group	Skupina, do které alarm patří	↓0 ↑9.22337E+18	Large (I64)
Desc	Popis alarmu	⊙Alarm Description	String

Výstup

iac	Kód aktuálního alarmu	Long (I32)
	0 signál v mezích	
	1 dolní alarm (LA) je aktivní	
	2 horní alarm (HA) je aktivní	
	4 druhý dolní alarm (LLA) je aktivní	
	8 druhý horní alarm (HHA) je aktivní	
	256 ... dolní alarm (LA) není potvrzen (NACK)	
	512 ... horní alarm (HA) není potvrzen (NACK)	
	1024 .. druhý dolní alarm (LLA) není potvrzen (NACK)	
	2048 .. druhý horní alarm (HHA) není potvrzen (NACK)	
	-1 neplatné meze alarmu	
E	Příznak chyby	Bool
	off ... bez chyby	
	on nastala chyba	
HHA	Indikátor druhého horního alarmu	Bool
HA	Indikátor horního alarmu	Bool
LA	Indikátor dolního alarmu	Bool
LLA	Indikátor druhého dolního alarmu	Bool
NACK	Indikátor nepotvrzeného alarmu	Bool

ARS – Uložení hodnoty do archivu

Symbol bloku

Licence: [STANDARD](#)



Popis funkce

Pokud je `RUN = on`, blok `ARS` zaznamená hodnotu ze vstupu `u` do archivu. Datový typ hodnoty na vstupu je specifikován parametrem `type` a stejný datový typ je použit i pro uložení v archivu. Parametr `subtype` umožňuje zadat typ alarmu stejně jako ho zapisují alarmové bloky `ALB`, `ALBI`, `ALN` a `ALNI`:

- **0: Low-alarm (LA)**: pro logickou hodnotu vyjadřuje indikaci sestupné hrany (`on→off`), pro číselnou hodnotu je to dolní alarm
- **1: High-alarm (HA)**: pro logickou hodnotu vyjadřuje indikaci náběžné hrany (`off→on`), pro číselnou hodnotu je to horní alarm
- **2: Second low-alarm (LLA)**: jedná se o druhý dolní alarm
- **3: Second high-alarm (HHA)**: jedná se o druhý horní alarm

Pro pole se parametr nepoužívá. Význam ostatních parametrů je stejný jako u ostatních bloků pro zápis do archivu.

Pokud je `type = Reference`, očekává se pole (sloupcový vektor nebo matice). Pokud je to matice, uloží se každý její sloupec jako samostatná úložka do archivu (tj. v jednom tiku tasku s tímto blokem vynikne v archivu tolik položek, kolik má matice sloupců).

Poznámka1: V případě polí, je archivní subsystém omezen na 255 hodnot v jedné úložce. Současně platí omezení na 512 byte dat v jedné úložce, takže pro typ `Short` se uloží nejvýše 128 hodnot, pro typ `Long` nejvýše 64 hodnot, 32 hodnot pro typ `Double`. Pokud je vstupní pole delší, blok uloží uvedené počty hodnot od začátku pole a nehlásí žádnou chybu.

Poznámka2: V případě `stringu` je archivní subsystém omezen na 65535 byte (znaků v UTF8 kódování může být méně). Pokud je vstupní text delší, blok uloží prvních 65535 byte od začátku pole a nehlásí žádnou chybu. Některé čtecí funkce mohou mít malý buffer a takto dlouhý text pak nelze vyčíst, doporučuje se proto nepřekračovat 4080 byte (znaků, pokud se používají jen znaky z anglická klávesnice).

Poznámka3: Parametr `id` obvykle slouží k provázání položky v archivu se zdrojovým blokem/signálem (a alarmem v některých případech). Proto se kontroluje jeho unikátnost v rámci celé konfigurace. Blok `ARS` je považován za nízkoúrovňový blok, který zapíše událost do archivu bez dalších souvislostí a kontrol. Proto se zde unikátnost parametru `id` nekontroluje. Pokud se například u binárního alarmu začnou v archivu objevovat

číselné nebo textové položky, generuje je téměř jistě nějaký blok **ARS** (nebo analogická funkce ve skriptu bloku [REXLANG](#)).

Tento blok nepropaguje kvalitu signálu. Více informací je uvedeno v sekci [1.4](#).

Vstup

u	Signál pro uložení do archivu	Any
RUN	Povolení běhu algoritmu	Bool

Parametr

type	Datový typ vstupního signálu	⊙12 Byte (U8)
	1 Bool	
	2 Byte (U8)	
	3 Short (I16)	
	4 Long (I32)	
	5 Word (U16)	
	6 DWord (U32)	
	7 Float (F32)	
	8 Double (F64)	
	9 Time	
	10 Large (I64)	
	11 Error	
	12 String	
	13 Reference	
arc	Seznam archivů pro zápis alarmů	Word (U16)
id	Unikátní identifikátor události v archivu	⊙1 Word (U16)
lvl	Úroveň (závažnost) alarmu	⊙1 Word (U16)
subtype	Typ alarmu	Byte (U8)
	0 dolní alarm (LA)	
	1 horní alarm (HA)	
	2 druhý dolní alarm (LLA)	
	3 druhý horní alarm (HHA)	

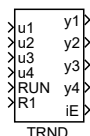
Výstup

iE	Kód chyby	Error

TRND – Záznam trendů v reálném čase

Symbol bloku

Licence: [STANDARD](#)



Popis funkce

Blok TRND slouží pro ukládání průběhů až čtyř vstupních signálů **u1** až **u4** do cyklických trendových bufferů v paměti cílového zařízení (target). Výhodou bloku TRND je synchronní ukládání dat s během exekutivy reálného času, které umožňuje ukládat do trendu i velmi rychlé signály. Na rozdíl od asynchronního ukládání dat na nadřazeném operátorském počítači (host) nedochází ke ztrátě některých vzorků nebo jejich vícenásobnému uložení. Data lze blokem TRND ukládat i pro velmi krátké periody spouštění úloh. Pro zvýšení přehlednosti lze jednotlivým trendům přiřadit specifické názvy, definovat jména a atributy signálů a také uchovávat konfiguraci jejich zobrazení. Tyto úpravy lze provádět v editoru po kliknutí na tlačítko **Configure**. Všechny změny jsou následně uloženy do parametrů **title** (název), **signals** (signály) a **view** (zobrazení).

Skutečný počet ukládaných průběhů určuje parametr **n**. V případě, že se trendové buffery s délkou 1 vzorků zaplní, začnou se přepisovat nejstarší vzorky. Do trendových bufferů se mohou ukládat data jednou za **pfac** spuštění bloku (decimace), tedy s periodou $pfac \cdot T_s$, kde T_s je perioda vykonávání bloku [s]. Ukládaná data mohou být zpracována podle hodnoty parametrů **ptype1** až **ptype4**. Další decimace s faktorem **afac** může být použita pro ukládání do archivů. Perioda ukládání pak je dána jako $afac \cdot pfac \cdot T_s$. Každá hodnota je ukládána s časovou značkou. Zdroj časové značky lze nastavit parametrem **timesrc** (více viz blok [TS](#)).

Seznam archivů pro zápis je specifikován parametrem **arc**, např. 1,3..5,8. Kód identifikace události v archivu **id** musí být unikátní v celém cílovém zařízení s řídicím systémem REXYGEN. Pro více informací o těchto parametrech viz úvod kapitoly 10.

Pro úsporu paměti na cílovém zařízení může být parametrem **btype** specifikován typ použitých trendových bufferů. Velikost paměti obsazená trendovými buffery je dána vztahem $s \cdot n \cdot 1$, kde s je velikost proměnné daného typu v bytech. Přednastavený typ **Double** zabírá 8 bytů na každý vzorek, pokud je tedy např. počet trendů $n = 4$, délka každého trendu $1 = 1000$, pak pro typ **Double** je zapotřebí $8 \cdot 4 \cdot 1000 = 32000$ bytů. V případě, že by byly vstupní signály měřeny z A/D převodníku s rozlišením do 16 bitů, mohly by být ukládány v typu **Word** s velikostí 2 byty na vzorek a velikost potřebné paměti by se zmenšila na jednu čtvrtinu. Velikosti jednotlivých datových typů a jejich rozsahy jsou uvedeny v tabulce 1.1 na straně 24.

Při použití jiného typu pro trendové buffery než je typ **Double** může nastat případ, že

se zpracovaná hodnota některého vstupu „nevejde“ do zvoleného typu bufferu a má hodnotu menší (větší) než je nejmenší (největší) zobrazitelné číslo v daném typu. V takovém případě se do bufferu uloží nejmenší (největší) zobrazitelné číslo v daném typu a chyba se binárně zakóduje do chybového výstupu **iE** podle následující tabulky (nepoužité bity jsou vypuštěny):

Chyba	Podkročení rozsahu				Překročení rozsahu			
Vstup	u4	u3	u2	u1	u4	u3	u2	u1
Číslo bitu	11	10	9	8	3	2	1	0
Váha bitu	2048	1024	512	256	8	4	2	1

V případě, že nastane najednou několik chyb, je výsledný chybový kód dán součtem vah jednotlivých chyb. Poznamenejme, že současné překročení a podkročení rozsahu na daném vstupu nemohou nastat zároveň.

Číst, zobrazovat a exportovat průběžně ukládaná data je možné v REXYGEN Studio ve **Watch** režimu. Po dvojkliku na příslušný TRND blok se otevře nová karta s předponou **Trend**.

POZOR: nastavení kteréhokoliv z parametrů **arc**, **afac**, **id** na 0 (prázdný) způsobí, že data se nezapisují do archivu a jsou dostupná jen v diagnostických nástrojích.

Tento blok nepropaguje kvalitu signálu. Více informací je uvedeno v sekci 1.4.

Vstup

u1..u4	Analogový vstupní signál	Double (F64)
RUN	Povolení běhu algoritmu	Bool
R1	Reset bloku	Bool

Parametr

n	Počet signálů (bufferů) v trendu	↓1 ↑4 ⊙4	Long (I32)
l	Počet vzorků pro každý buffer trendu	↓0 ↑268435000 ⊙1000	Long (I32)
btype	Typ všech použitých bufferů	⊙8	Long (I32)
	1 Bool		
	2 Byte (U8)		
	3 Short (I16)		
	4 Long (I32)		
	5 Word (U16)		
	6 DWord (U32)		
	7 Float (F32)		
	8 Double (F64)		
	10 Large (I64)		

<code>p1..p4</code>	Zpracování dat	⊙1	Long (I32)
1	Uložit		
2	Minimum		
3	Maximum		
4	Součet		
5	Průměr		
6	Směrodatná odchylka		
7	Rozptyl		
<code>pfac</code>	Faktor zpracování dat	↓1 ↑1000000 ⊙1	Long (I32)
<code>afac</code>	Archivační faktor	↓0 ↑1000000	Long (I32)
<code>arc</code>	Seznam archivů pro zápis alarmů		Word (U16)
<code>id</code>	Unikátní identifikátor události v archivu	⊙1	Word (U16)
<code>title</code>	Název trendu	⊙Trend Title	String
<code>timesrc</code>	Zdroj časových značek	⊙1	Long (I32)
1	CORETIMER		
2	CORETIMER (přesný)		
3	RTC (UTC)		
4	RTC (lokální čas)		
5	PFC		
6	TSC		
<code>signals</code>	Názvy a atributy signálů ve formátu CSV		String
<code>view</code>	Uložená konfigurace pohledu		String

Výstup

<code>y1..y4</code>	Analogový výstupní signál	Double (F64)
<code>iE</code>	Kód chyby ukládání (po bitech)	Long (I32)

TRNDV – Záznam trendů v reálném čase (pro vektory)

Symbol bloku

Licence: [STANDARD](#)



Popis funkce

Blok TRNDV je velmi podobný bloku [TRND](#). Umožňuje však zaznamenávat více než 4 signály. Signály jsou předávány na vstup **uVec** formou vektoru. Počet zpracovávaných signálů je pak dán parametrem **n**. Na rozdíl od [TRND](#) je pro zastavení vykovávání bloku nutné nastavit vstup **HLD** na **on**.

Tento blok nepropaguje kvalitu signálu. Více informací je uvedeno v sekci [1.4](#).

Vstup

uVec	Vektorový signál určený k uložení	Reference
HLD	Pozastavení	Bool
R1	Reset bloku	Bool

Parametr

n	Počet signálů (bufferů) v trendu	↓1 ↑64 ⊙8	Long (I32)
l	Počet vzorků pro každý buffer trendu	↓2 ↑268435000 ⊙1000	Long (I32)
btype	Typ všech použitých bufferů	⊙8	Long (I32)
	1 Bool		
	2 Byte (U8)		
	3 Short (I16)		
	4 Long (I32)		
	5 Word (U16)		
	6 DWord (U32)		
	7 Float (F32)		
	8 Double (F64)		
	10 Large (I64)		
pfac	Faktor zpracování dat	↓1 ↑1000000 ⊙1	Long (I32)
afac	Archivační faktor	↓0 ↑1000000	Long (I32)
arc	Seznam archivů pro zápis alarmů		Word (U16)
id	Unikátní identifikátor události v archivu	⊙1	Word (U16)
title	Název trendu	⊙Trend Title	String

timesrc	Zdroj časových značek	⊙1 Long (I32)
	1 CORETIMER	
	2 CORETIMER (přesný)	
	3 RTC (UTC)	
	4 RTC (lokální čas)	
	5 PFC	
	6 TSC	
signals	Názvy a atributy signálů ve formátu CSV	String
view	Uložená konfigurace pohledu	String

Výstup

iE	Kód chyby	Error
	i kód chyby REXYGENu	

Kapitola 11

STRING – Bloky pro práci s řetězcí

Obsah

CNS – Textová konstanta	362
CONCAT – Spojení stringů (podle vzoru)	363
FIND – Nalezení textu	364
ITOS – Konverze celého čísla na text	365
LEN – Délka textu	366
MID – Výřez textu	367
PJROCT – Získání číselných hodnot z textu ve formátu JSON	368
PJSEXOCT – Získání textových hodnot z textu ve formátu JSON	370
PJSOCT – Získání textových hodnot z textu ve formátu JSON	371
REGEXP – Parser regulárních výrazů	373
REPLACE – Náhrada textu	376
RTOS – Konverze čísla na text	377
SELSOCT – Výběr textu z několika vstupů	378
STOR – Konverze textu na číslo	380
TRIM – Odstranění mezer na začátku a konci textu	381

Knihovna **STRING** je věnována manipulaci a analýze řetězců v systému **REXYGEN**. Zahrnuje bloky jako **CONCAT** pro zřetězení řetězců, **FIND** pro vyhledávání v řetězcích a **REPLACE** pro nahrazování segmentů řetězců. Knihovna nabízí **LEN** a **MID** pro určení délky řetězce a extrakci podřetězců. Pokročilé porovnávání vzorů poskytuje **REGEXP**. Konverzní bloky jako **ITOS**, **STOR** a **RTOS** převádějí celá a reálná čísla na řetězce, zatímco jednoduchý blok **CNS** definuje řetězcovou konstantu. Knihovna navíc obsahuje bloky jako **PJROCT** pro analýzu JSON. Tato kolekce bloků je nezbytná pro manipulaci a zpracování řetězcových dat v různých aplikacích.

CNS – Textová konstanta

Symbol bloku

Licence: [STANDARD](#)



Popis funkce

Blok **CNS** je jednoduchá konstanta řetězce s maximální dostupnou velikostí. Hodnota **scv** je vždy oříznuta na **nmax**. Pokud není prázdný řetězcový parametr **filename**, načte inicializační data ze souboru **filename** na hostitelském počítači.

Tento blok propaguje kvalitu signálu. Více informací je uvedeno v sekci [1.4](#).

Parametr

scv	Textová hodnota		String
nmax	Rezervovaná paměť pro řetězec	↓0 ↑65520	Long (I32)
filename	Datový soubor (obsah je nahrán do scv, pokud je nastaven)		String

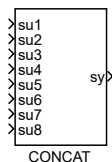
Výstup

sy	Výstupní textová hodnota	String

CONCAT – Spojení stringů (podle vzoru)

Symbol bloku

Licence: [STANDARD](#)



Popis funkce

Blok **CONCAT** spojuje až 8 vstupních řetězců **su1** až **su8** podle vzoru specifikovaného v parametru **ptrn**.

Tento blok propaguje kvalitu signálu. Více informací je uvedeno v sekci [1.4](#).

Vstup

su1 . . su8	Vstupní textová hodnota	String

Parametr

ptrn	Spojovací vzor	⊙%1%2%3%4	String
nmax	Rezervovaná paměť pro řetězec	↓0 ↑65520	Long (I32)

Výstup

sy	Výstupní textová hodnota	String

FIND – Nalezení textu

Symbol bloku

Licence: [STANDARD](#)



Popis funkce

Blok **FIND** vyhledává řetězec **su2** v řetězci **su1**. Pokud je **su2** nalezen, na výstupu **pos** je vrácen index prvního výskytu **su2** v **su1** (počítáno od jedničky). Pokud **su2** není nalezen, je na výstupu **pos** nula a na výstupu **iE** indikována chyba. Oba vstupní řetězce jsou oříznuty na délku **nmax**.

Tento blok propaguje kvalitu signálu. Více informací je uvedeno v sekci [1.4](#).

Vstup

su1	Vstupní textová hodnota	String
su2	Vstupní textová hodnota	String

Parametr

nmax	Rezervovaná paměť pro řetězec	↓0 ↑65520	Long (I32)

Výstup

pos	Poloha hledaného textu	Long (I32)
iE	Kód chyby	Error

ITOS – Konverze celého čísla na text

Symbol bloku

Licence: [STANDARD](#)



Popis funkce

Blok **ITOS** slouží k převedení celého čísla na text. Parametr **len** určuje minimální délku výstupního řetězce. Pokud má číslo menší počet číslic, budou podle parametru **mode** doplněny nuly nebo mezery. Parametr **radix** určuje číselnou soustavu, ve které se má převod provést. Výstupní řetězec neobsahuje žádnou identifikaci použité číselné soustavy (např. předponu 0x u šestnáctkové soustavy).

Tento blok propaguje kvalitu signálu. Více informací je uvedeno v sekci [1.4](#).

Vstup

n	Celočíselný vstupní signál	Long (I32)

Parametr

len	Minimální délka výstupního řetězce	↓0 ↑30	Long (I32)
mode	Formát výstupního textu	⊙1	Long (I32)
	1 zarovnat vpravo, vyplnit mezerami		
	2 zarovnat vpravo, vyplnit nulami		
	3 zarovnat vlevo, vyplnit mezerami		
radix	Číselná soustava	⊙10	Long (I32)
	2 dvojková		
	8 osmičková		
	10 desítková		
	16 šestnáctková		

Výstup

sy	Výstupní textová hodnota	String

LEN – Délka textu

Symbol bloku

Licence: [STANDARD](#)



Popis funkce

Blok LEN vrací aktuální délku řetězce v **su** v UTF-8 znacích.

Tento blok propaguje kvalitu signálu. Více informací je uvedeno v sekci [1.4](#).

Vstup

su	Vstupní textová hodnota	String

Parametr

nmax	Rezervovaná paměť pro řetězec	↓0 ↑65520	Long (I32)

Výstup

len	Délka vstupního textu	Long (I32)

MID – Výřez textu

Symbol bloku

Licence: [STANDARD](#)



Popis funkce

Blok MID extrahuje podřetězec **sy** z řetězce **su**. Vstupy **l** a **p** specifikují pozici a délku řetězce, který se má extrahovat v UTF-8 znacích. Hodnota vstupu **p** je počítána od jedné.

Tento blok propaguje kvalitu signálu. Více informací je uvedeno v sekci [1.4](#).

Vstup

su	Vstupní textová hodnota	String
l	Délka výstupního textu	Long (I32)
p	Pozice výstupního textu	Long (I32)

Parametr

nmax	Rezervovaná paměť pro řetězec	↓0 ↑65520	Long (I32)

Výstup

sy	Výstupní textová hodnota	String
iE	Kód chyby	Error

PJROCT – Získání číselných hodnot z textu ve formátu JSON

Symbol bloku

Licence: [STANDARD](#)

Popis funkce

Blok PJROCT parsuje vstupní JSON řetězec `jtxt` podle zadaných parametrů `name*` pokud vstup `RUN` má hodnotu `on`. Výstupní signály jsou typu `real`. V případě chyby je na výstupy `y*` vložena náhradní hodnota z parametru `yerr` (např. zadaný objekt neexistuje nebo hodnota není číslo).

Blok na vstupu `jtxt` očekává text ve formátu JSON. Na výstupech `y1` až `y7` jsou pak po řadě hodnoty objektů identifikovaných parametry `name1` až `name7`. Pokud je některý z parametrů `name1` až `name7` prázdný, bude prázdný i příslušný výstup a není to považováno za chybu. Blok vstupní string vyhodnocuje jen pokud je `RUN = on`. Na výstupu `iE` je indikována chyba. Mohou nastávat tyto případy:

- **0** – bez chyby
- **-1** – některý z parametrů `name1` až `name7` odkazuje na objekt, který se ve vstupním textu (na vstupu `jtxt`) nevyskytuje
- **-103** – textu na vstupu `jtxt` neodpovídá JSON formátu
- **-106** – všechny z parametrů `name1` až `name7` odkazují na objekt, který se ve vstupním textu (na vstupu `jtxt`) nevyskytuje

Příklady

Předpokládejme

```
jtxt = '{"id": 12345, "params":{"temperature": 23, "pressure": 2.34},
"description":"reactor1","values":[12, 34.5, 45.0, 30.2]}'
name1 = "params.temperature",
name2 = "values[0]",
name3 = "pressure",
name4 = "description",
```

pak na výstupu `sy1` bude hodnota `"23"`, na výstupu `y2` bude hodnota `"12"`, na výstupu `y3` bude hodnota parametru `yerr` a bude signalizována chyba, na výstupu `y4` bude hodnota parametru `yerr` a bude signalizována chyba.

Tento blok nepropaguje kvalitu signálu. Více informací je uvedeno v sekci [1.4](#).

Vstup

<code>jtxt</code>	Text v JSON formátu	String
<code>RUN</code>	Povolení běhu algoritmu	Bool

Parametr

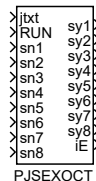
<code>name1..name8</code>	Jméno objektu v textu formátu JSON	String
<code>nmax</code>	Rezervovaná paměť pro řetězec	↓0 ↑65520 Long (I32)
<code>yerr</code>	Náhradní hodnota pro případ chyby	Double (F64)

Výstup

<code>y1..y8</code>	Výstup bloku	Double (F64)
<code>iE</code>	Kód chyby	Error

PJSEXOCT – Získání textových hodnot z textu ve formátu JSON

Symbol bloku

Licence: [STANDARD](#)

Popis funkce

Blok PJSEXOCT je téměř shodný s blokem [PJSOCT](#). Na vstupu `jtxt` očekává text ve formátu JSON. Na výstupech `sy1` až `sy7` jsou pak po řadě hodnoty objektů identifikovaných parametry `name1` až `name7`. Na rozdíl od bloku [PJSOCT](#) mohou parametry `name1` až `name7` obsahovat zástupný znak `%` + číslo místo kterého se dosadí text ze vstupu `sn` + číslo.

Příklady

Předpokládejme

```
sn1 = "2",
sn2 = "rpm",
name1 = "motor[%1].temp",
name2 = "motor[%1].%2",
```

pak na výstupu `sy1` bude hodnota objektu `motor[2].temp`, a na výstupu `sy2` bude hodnota objektu `motor[2].rpm`.

Tento blok nepropaguje kvalitu signálu. Více informací je uvedeno v sekci [1.4](#).

Vstup

<code>jtxt</code>	Text v JSON formátu	String
<code>RUN</code>	Povolení běhu algoritmu	Bool
<code>sn1..sn8</code>	Jméno objektu v textu formátu JSON	String

Parametr

<code>name1..name8</code>	Jméno objektu v textu formátu JSON	String
<code>nmax</code>	Rezervovaná paměť pro řetězec	↓0 ↑65520 Long (I32)

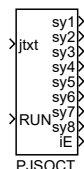
Výstup

<code>sy1..sy8</code>	Výstupní textová hodnota	String
<code>iE</code>	Kód chyby	Error

PJSOCT – Získání textových hodnot z textu ve formátu JSON

Symbol bloku

Licence: [STANDARD](#)



Popis funkce

Blok **PJSOCT** parsuje vstupní JSON řetězec **jtxt** podle zadaných parametrů **name*** pokud vstup **RUN** má hodnotu **on**. Výstupní signály jsou typu **string**.

Tento blok na vstupu **jtxt** očekává text ve formátu JSON. Na výstupech **sy1** až **sy7** jsou pak po řadě hodnoty objektů identifikovaných parametry **name1** až **name7**. Pokud je některý z parametrů **name1** až **name7** prázdný, bude prázdný i příslušný výstup a není to považováno za chybu. Blok vstupní string vyhodnocuje jen pokud je **RUN = on**. Na výstupu **ie** je indikována chyba. Mohou nastávat tyto případy:

- **0** – bez chyby
- **-1** – některý z parametrů **name1** až **name7** odkazuje na objekt, který se ve vstupním textu (na vstupu **jtxt**) nevyskytuje
- **-103** – textu na vstupu **jtxt** neodpovídá JSON formátu
- **-106** – všechny z parametrů **name1** až **name7** odkazují na objekt, který se ve vstupním textu (na vstupu **jtxt**) nevyskytuje

Příklady

Předpokládejme

```
jtxt = "{\"id\": 12345, \"params\":{\"temperature\": 23, \"pressure\": 2.34},
\"description\":\"reactor1\",\"values\":[12, 34.5, 45.0, 30.2]}\",
name1 = \"params.temperature\",
name2 = \"values[0]\",
name3 = \"pressure\",
name4 = \"description\",
```

pak na výstupu **sy1** bude hodnota **"23"**, na výstupu **sy2** bude hodnota **"12"**, výstup **sy3** zůstane prázdný a bude signalizována chyba, na výstupu **sy4** bude hodnota **"reactor1"**.

Tento blok nepropaguje kvalitu signálu. Více informací je uvedeno v sekci [1.4](#).

Vstup

jtxt	Text v JSON formátu	String
RUN	Povolení běhu algoritmu	Bool

Parametr

name1..name8	Jméno objektu v textu formátu JSON	String
nmax	Rezervovaná paměť pro řetězec	↓0 ↑65520 Long (I32)

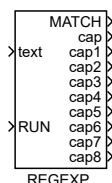
Výstup

sy1..sy8	Výstupní textová hodnota	String
iE	Kód chyby	Error

REGEXP – Parser regulárních výrazů

Symbol bloku

Licence: [ADVANCED](#)



Popis funkce

Blok **REGEXP** implementuje nejběžnější podmnožinu regulárních výrazů, jak je známe například z příkazu **regex** v Perl nebo **C#**, či z příkazu **grep** známého z příkazové řádky operačních systémů Unix.

Podporovaná syntaxe je následující:

- **(?i)** ... Musí být na začátku regexu. Způsobuje porovnávání bez rozlišení velkých a malých písmen.
- **^** ... Vyhledání začátku bufferu.
- **\$** ... Vyhledání konce bufferu.
- **()** ... Seskupování a zachytávání podřetězců.
- **\s** ... Vyhledání bílého znaku.
- **\S** ... Vyhledání znaku, který není bílý.
- **\d** ... Vyhledání desetinné číslce.
- **\n** ... Vyhledání znaku nového řádku.
- **\r** ... Vyhledání znaku návratu vozíku.
- **\f** ... Vyhledání znaku formátování stránky.
- **\v** ... Vyhledání znaku vertikální tabulace.
- **\t** ... Vyhledání znaku horizontální tabulace.
- **\b** ... Vyhledání znaku backspace.
- **+** ... Vyhledání jednou nebo vícekrát (chamtivě).
- **+** ... Vyhledání jednou nebo vícekrát (nechamtivě).

- ... Vyhledání nula nebo vícekrát (chamtivě).
- ? ... Vyhledání nula nebo vícekrát (nechamtivě).
- ? ... Vyhledání nula nebo jednou (nechamtivě).
- `x|y` ... Vyhledání `x` nebo `y` (operační alternativa).
- `\meta` ... Vyhledání jednoho z metaznaků: `^$().[*+?|\`
- `\xHH` ... Vyhledání bajtu s hexadecimální hodnotou `0xHH`, např. `\x4a`.
- `[...]` ... Vyhledání jakéhokoli znaku ze sady. Jsou podporovány rozsahy jako `[a-z]`.
- `[^...]` ... Vyhledání jakéhokoli znaku kromě těch ve sadě. Jsou podporovány rozsahy jako `[a-z]`.

Příklady

- `[0-9]+` ... Najde první celé číslo ve vstupním řetězci (a vloží jej do výstupu `cap`).
- `[-+]?[0-9]*\.[0-9]+([eE] [-+]?[0-9]+)?` ... Najde první reálné číslo ve vstupním řetězci (a vloží jej do výstupu `cap`).
- `^\s*(.*?)\s*$` ... Vloží ořezaný vstupní řetězec do výstupu `cap1`.
- `num\s*:\s*([0-9]*\.[0-9]*)` ... Očekává vstupní řetězec ve formátu JSON; najde parametr `num`, a jeho hodnotu vloží do výstupu `cap1`.

Tento blok nepropaguje kvalitu signálu. Více informací je uvedeno v sekci 1.4.

Vstup

<code>text</code>	Text k rozpoznání	<code>String</code>
<code>RUN</code>	Povolení běhu algoritmu	<code>Bool</code>

Parametr

<code>expr</code>	Regulární výraz k rozpoznání	<code>String</code>
<code>nmax</code>	Rezervovaná paměť pro řetězec	↓0 ↑65534 <code>Long (I32)</code>
<code>bufmax</code>	Velikost intrní pracovní paměti (0 = automaticky)	↓0 ↑10000000 <code>Long (I32)</code>

Výstup

<code>MATCH</code>	Příznak rozpoznání	<code>Bool</code>
<code>cap</code>	Rozpoznaný text	<code>String</code>

cap1..cap8 Rozpoznaný text

String

REPLACE – Náhrada textu

Symbol bloku

Licence: [STANDARD](#)



Popis funkce

Blok **REPLACE** nahrazuje podřetězec z **su1** řetězcem **su2** a výsledek ukládá do **sy**. Parametry **l** a **p** určují pozici a délku nahrazovaného řetězce v UTF-8 znacích. Parametr **p** je číslován od jedné.

Tento blok propaguje kvalitu signálu. Více informací je uvedeno v sekci [1.4](#).

Vstup

su1	Vstupní textová hodnota	String
su2	Vstupní textová hodnota	String
l	Délka původního textu	Long (I32)
p	Pozice původního textu	Long (I32)

Parametr

nmax	Rezervovaná paměť pro řetězec	↓0 ↑65520	Long (I32)

Výstup

sy	Výstupní textová hodnota	String
iE	Kód chyby	Error

RTOS – Konverze čísla na text

Symbol bloku

Licence: [STANDARD](#)



Popis funkce

Blok **RTOS** slouží ke konverzi desetinného čísla ze vstupu **u** na výstupní řetězec **su**. Přesnost a formát převodu určují parametry **prec** a **mode**. Možné hodnoty parametru **mode** jsou:

- **1: Nejlepší přizpůsobení** – pevný desetinný formát, ale pro extrémně malá nebo velká čísla exponenciální formát; parametr **prec** je celkový maximální počet znaků ve výstupu (mantisa pro exponenciální formát)
- **2: Normální** – pevný desetinný formát; parametr **prec** je počet desetinných míst
- **3: Exponenciální** – vědecký formát; parametr **prec** je počet desetinných míst

Tento blok propaguje kvalitu signálu. Více informací je uvedeno v sekci [1.4](#).

Vstup

u	Analogový vstupní signál	Double (F64)

Parametr

prec	Přesnost (počet cifer)	↓0 ↑20	Long (I32)
mode	Formát výstupního textu	⊙1	Long (I32)
	1 nejvhodnější		
	2 normální		
	3 exponenciální		

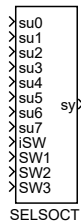
Výstup

sy	Výstupní textová hodnota	String

SELSOCT – Výběr textu z několika vstupů

Symbol bloku

Licence: [STANDARD](#)



Popis funkce

Blok **SEL**OCT vybere jeden ze vstupních řetězců a zkopíruje ho do výstupního řetězce **sy**. Výběr aktivního signálu **u0**...**u15** je založen na vstupu **iSW** nebo na binárních vstupech **SW1**...**SW3**. Tyto dva režimy jsou rozlišeny pomocí binárního příznaku **BINF**. Signál je vybrán podle následující tabulky:

iSW	SW1	SW2	SW3	y
0	off	off	off	u0
1	on	off	off	u1
2	off	on	off	u2
3	on	on	off	u3
4	off	off	on	u4
5	on	off	on	u5
6	off	on	on	u6
7	on	on	on	u7

Tento blok propaguje kvalitu signálu. Více informací je uvedeno v sekci [1.4](#).

Vstup

su0...su7	Vstupní textová hodnota	String
iSW	Selektor aktivního signálu	Long (I32)
SW1	Binární vstup pro výběr	Bool
SW2	Binární vstup pro výběr	Bool
SW3	Binární vstup pro výběr	Bool

Parametr

BINF	Výběr pomocí binárních vstupů	Bool
nmax	Rezervovaná paměť pro řetězec	↓0 ↑65520 Long (I32)

Výstup

sy

Zvolený vstupní signál

String

STOR – Konverze textu na číslo

Symbol bloku

Licence: **STANDARD**



Popis funkce

Blok **STOR** převede řetězec na vstupu **su** na reálné číslo na výstupu **y**. Pokud se převod nezdaří, je indikována chyba v **E**.

Tento blok propaguje kvalitu signálu. Více informací je uvedeno v sekci [1.4](#).

Vstup

su	Vstupní textová hodnota	String

Parametr

yerr	Náhradní hodnota pro případ chyby	Double (F64)

Výstup

y	Analogový výstupní signál	Double (F64)
E	Příznak chyby	Bool

TRIM – Odstranění mezer na začátku a konci textu

Symbol bloku

Licence: [STANDARD](#)



Popis funkce

Blok **TRIM** odstraní z vstupního řetězce **su** vedoucí a koncové bílé znaky a výsledek uloží do výstupního řetězce **sy**.

Tento blok propaguje kvalitu signálu. Více informací je uvedeno v sekci [1.4](#).

Vstup

su	Vstupní textová hodnota	String

Parametr

nmax	Rezervovaná paměť pro řetězec	↓0 ↑65520	Long (I32)

Výstup

sy	Výstupní textová hodnota	String

Kapitola 12

PARAM – Bloky pro manipulaci s parametry

Obsah

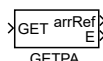
GETPA – Blok pro vzdálené získání vektorového parametru	384
GETPB, GETPI, GETPR – Bloky pro vzdálené získání parametru . . .	386
GETPS – Blok pro vzdálené získání parametru typu string	388
GETPX – Blok pro vzdálené získání parametru	389
PARA – Blok s vektorovým parametrem nastavitelným ze vstupu	391
PARE – Blok s parametrem výběr ze seznamu nastavitelným ze vstupu	393
PARB, PARI, PARR – Bloky s nastavitelným parametrem ze vstupu	394
PARS – Blok s parametrem typu string nastavitelným ze vstupu	396
PARX – Blok s parametrem nastavitelným ze vstupu	397
SETPA – Blok pro vzdálené nastavování vektorového parametru .	399
SETPB, SETPI, SETPR – Bloky pro vzdálené nastavování parametru	401
SETPS – Blok pro vzdálené nastavování parametru typu string .	404
SETPX – Blok pro vzdálené nastavování parametru	405
SGSLP – Nastavování, čtení, ukládání a načítání parametrů	407
SILO – Uložení vstupního signálu, načtení výstupního signálu . .	411
SILOS – Uložení vstupního řetězce, načtení výstupního řetězce .	413

Knihovna PARAM je určena pro správu parametrů a zpracování signálů v systému REXYGEN. Obsahuje bloky jako **PARR** a jeho varianty pro definování a úpravu různých typů parametrů. Bloky pro čtení parametrů jiných bloků jako **GETPA** a **GETPS**. Naopak **SETPA**, **SETPR** a **SETPS** se používají pro dynamické nastavení hodnot parametrů jiných bloků. Knihovna navíc obsahuje **SILO** a **SILOS** pro export a import hodnot ze souboru. Tato knihovna je klíčová pro systémy vyžadující dynamickou manipulaci s parametry a možnosti čtení/ukládání hodnot do souboru.

GETPA – Blok pro vzdálené získání vektorového parametru

Symbol bloku

Licence: [STANDARD](#)



Popis funkce

Blok **GETPA** slouží ke vzdálenému získávání vektorových parametrů ostatních bloků v modelu. Může pracovat ve dvou režimech, které se přepínají parametrem **GETF**. Pro **GETF = off** je na výstup **arrRef** vyveden vzdálený vektorový parametr při startu a dále pak při každé změně sledovaného vzdáleného parametru. Jestliže parametr **GETF** je **on**, pak bloky pracují v režimu jednorázového čtení vzdáleného parametru, který se přečte vždy, když nastane náběžná hrana (**off**→**on**) na vstupu **GET**.

Jméno vzdáleného parametru určuje textový parametr **sc** (string connection), který se zadává ve tvaru `<cesta_k_bloku:jmeno_parametru>`. Cesta k bloku, jehož parametr má být získán, může obsahovat tečkami oddělené hierarchické úrovně, na jejichž konci je název bloku a může být:

- **Relativní:**

- Začíná znakem '.', tedy v úrovni, do které je umístěn blok **GETPA**. Příklady cest: `".CNDR:yp"`, `".Lights.ATMT:touts"`.
- Začíná znaky '..', tedy v úrovni nad blokem **GETPA**. Příklady: `". .CNDR:yp"`, `". .Lights.ATMT:touts"`.

- **Relativní k tasku:** Začíná v základní úrovni tasku, do které je umístěn daný blok **GETPA**. V tomto případě text začíná znakem '%'. Příklady hodnot cest: `"%CNDR:yp"`, `"%Lights.ATMT:touts"`.

- **Absolutní:** Úplná posloupnost hierarchických úrovní až k požadovanému bloku. V případě, že má být čten parametr z bloku umístěného v úloze ovladače (pro konfiguraci viz. blok [IOTASK](#)), je v první úrovni hierarchie uveden znak '&' následovaný názvem ovladače. Příklady hodnot absolutních cest: `"uloha1.vstupy.ATMT:touts"`, `"&EfaDrv.mereni.CNDR:yp"`.

Pořadí a názvy jednotlivých hierarchických úrovní jsou zobrazeny ve stromové struktuře konfigurace v diagnostice programu **REXYGEN Studio**.

Varování: Pokud se čte hodnota pole z jiné úlohy, je pro zajištění konzistence hodnot nutné použít tzv. semafor a počkat na dokončení úlohy, ze které se hodnota pole čte. Po celou dobu čekání na dokončení je úloha s blokem **GETPA** pozastavena! Z praktického hlediska to znamená, že blok **GETPA** se musí umístit do úlohy, která trvá dlouho a číst hodnotu pole z úlohy, která trvá krátce. Pokud je to opačně, dochází k čekání rychlé

úlohy na pomalejší úlohu a rychlejší úloha se tím zpozdí. V této situaci je vhodné použít blok [SETPA](#) v déletrvající úloze.

Poznámka 1: Pokud je parametr `GETF = off` a navíc zdrojové pole (určené parametrem `sc`) je ze stejného tasku jako blok `GETPA`, na výstup se dává přímý odkaz na původní pole. To šetří paměť i procesorový čas. V tomto případě jsou parametry `nmax`, `etype` ignorovány.

Poznámka 2: Pokud se použije více bloků `GETPA` pro čtení polí v jiném tasku, není zajištěno, že se všechny pole přečtou v jedné periodě druhé úlohy. Je pouze zajištěno, že dříve provedený blok `GETPA` přečte pole ze stejné nebo dřívější periody druhého tasku než později provedený blok `GETPA`. Pořadí spouštění je vidět v diagnostice programu REXYGEN Studio.

Poznámka 3: Vzdálené pole (ze kterého se čtou data) může být jak primární pole (např. `CNA:acn`, `RTOV:xVec`, `MX_MAT:ay`), tak reference na pole (např. `CNA:vec`, `RTOV:yVec`, `SUBSYSTEM:Outport`). Reference na pole by měla být umístěna ve stejném tasku, jako primární pole, jinak hrozí nekonzistence.

Poznámka 4: Pro čtení hodnoty vzdáleně lze použít také blok [INCONN](#).

Tento blok propaguje kvalitu signálu. Více informací je uvedeno v sekci [1.4](#).

Vstup

GET	Vstup pro jednorázové přečtení parametru	Bool

Parametr

<code>sc</code>	Jméno vzdáleného parametru	String
<code>GETF</code>	Načtení parametru pouze po vyžádání	Bool
	<code>off ...</code> režim průběžného čtení parametru	
	<code>on</code> režim jednorázového přečtení parametru	
<code>nmax</code>	Maximální velikost pole	↓10 ⊙256 Long (I32)
<code>etype</code>	Typ prvků	⊙8 Long (I32)
	1 Bool	
	2 Byte (U8)	
	3 Short (I16)	
	4 Long (I32)	
	5 Word (U16)	
	6 DWord (U32)	
	7 Float (F32)	
	8 Double (F64)	
	10 Large (I64)	

Výstup

<code>arrRef</code>	Odkaz na pole (vektor nebo matice)	Reference
<code>E</code>	Příznak chyby	Bool
	<code>off ...</code> bez chyby	
	<code>on</code> nastala chyba	

GETPB, GETPI, GETPR – Bloky pro vzdálené získání parametru

Symbody bloků

Licence: [STANDARD](#)



Popis funkce

Bloky **GETPR**, **GETPI** a **GETPB** slouží pro vzdálené získávání parametrů ostatních bloků v projektu. Bloky mají identickou funkci, liší se pouze v typu parametru, který získávají. Blok **GETPR** je pro reálné číslo, **GETPI** pro celé číslo a **GETPB** pro Booleovskou hodnotu. Kvůli dodržení konvence v pojmenovávání proměnných [1.3](#) se výstupy jednotlivých bloků nazývají podle typu získávaného parametru:

- **y** – reálný výstup bloku **GETPR**,
- **k** – celočíselný výstup **GETPI**,
- **Y** – Boolean výstup pro **GETPB**.

Bloky mohou pracovat ve dvou režimech, které se přepínají parametrem **GETF**. Pro **GETF = off** je hodnota výstupu **y** (nebo **k**, **Y**) nastavena na hodnotu vzdáleného parametru při startu a dále pak při každé změně sledovaného vzdáleného parametru. Jestliže parametr **GETF** je **on**, pak bloky pracují v režimu jednorázového čtení vzdáleného parametru, který se přečte vždy, když nastane náběžná hrana (**off**→**on**) na vstupu **GET**.

Jméno vzdáleného parametru určuje textový parametr **sc** (string connection), který se zadává ve tvaru `<cesta_k_bloku:jmeno_parametru>`. Rovněž je možné přistupovat k jednotlivým prvkům parametrů typu pole (např. parametr **tout** bloku [ATMT](#)). Toho se dosáhne pomocí hranatých závorek a čísla prvku, např. tedy `.ATMT:touts[2]`, číslování je od 0, uvedený propojovací řetězec tedy odkazuje na třetí prvek pole.

Cesta k bloku, jehož parametr má být čten, může obsahovat hierarchické úrovně oddělené tečkami následovanými názvem bloku. Cesta může být relativní nebo absolutní:

- **Relativní::**
 - Začíná znakem '.', tedy v úrovni, do které je umístěn blok **GETPR** (nebo **GETPI**, **GETPB**). Příklady cest: `".GAIN:k"`, `".Motor1.Poloha:ycn"`.
 - Začíná znaky '..', tedy v úrovni nad blokem **GETPR** (nebo **GETPI**, **GETPB**). Příklady cest: `"..GAIN:k"`, `"..Motor1.Poloha:ycn"`.
- **Relativní k tasku:** Začíná v základní úrovni tasku, do které je umístěn daný blok **GETPR** (nebo **GETPI**, **GETPB**). V tomto případě text začíná znakem '%'. Příklady hodnot cest: `"%GAIN:k"`, `"%Motor1.Poloha:ycn"`.

- **Absolutní:** Úplná posloupnost hierarchických úrovní až k požadovanému bloku. V případě, že má být čten parametr z bloku umístěného v úloze ovladače (pro konfiguraci viz. blok [IOTASK](#)), je v první úrovni hierarchie uveden znak '&' následovaný názvem ovladače. Příklady hodnot absolutních cest: "uloha1.vstupy.lin1:u2", "&EfaDrv.mereni.DER1:n".

Pořadí a názvy jednotlivých hierarchických úrovní jsou zobrazeny ve stromové struktuře v sekci **Diagnostics** programu REXYGEN Studio.

Varování: Pokud se čte hodnota z jiné úlohy, je pro zajištění konzistence hodnoty nutné použít tzv. semafor a počkat na dokončení úlohy, ze které se hodnota čte. Po celou dobu čekání na dokončení je úloha s blokem **GETPx** pozastavena! Z praktického hlediska to znamená, že blok **GETPx** se musí umístit do úlohy, která trvá dlouho a číst hodnotu z úlohy, která trvá krátce. Pokud je to opačně, dochází k čekání rychlé úlohy na pomalejší úlohu a rychlejší úloha se tím zpozdí. V této situaci je vhodné použít blok **SETPx** v děletrvajícím úloze.

Poznámka 1: Pokud se použije více bloků **GETPx** pro čtení dat v jiném tasku, není zajištěno, že se všechny hodnoty přečtou v jedné periodě druhé úlohy. Je pouze zajištěno, že dříve provedený blok **GETPx** přečte hodnotu ze stejné nebo dřívější periody druhého tasku než později provedený blok **SETPx**. Pořadí spouštění je vidět v diagnostice programu REXYGEN Studio. Pokud je důležité přečíst všechny hodnoty ve stejné periodě, musí se přenášet pomocí bloků [Inport](#) a [Outport](#) nebo z hodnot vytvořit pole a číst je najednou blokem [GETPA](#).

Poznámka 2: Pro vzdálené čtení hodnoty lze použít i bloky [GETPX](#) a [INCONN](#).

Tento blok propaguje kvalitu signálu. Více informací je uvedeno v sekci [1.4](#).

Vstup

GET	Vstup pro jednorázové přečtení parametru	Bool

Parametr

sc	Jméno vzdáleného parametru	String
GETF	Načtení parametru pouze po vyžádání	Bool
	off ... režim průběžného čtení parametru	
	on režim jednorázového přečtení parametru	

Výstup

y	Hodnota parametru	Double (F64)
E	Příznak chyby	Bool
	off ... bez chyby	
	on nastala chyba	

GETPS – Blok pro vzdálené získání parametru typu string

Symbol bloku

Licence: [STANDARD](#)



Popis funkce

Blok **GETPS** má stejnou funkci jako bloky [GETPR](#), [GETPI](#) a [GETPB](#), liší se pouze v tom, že získává hodnotu parametru typu string (řetězec).

Poznámka: Ke čtení hodnoty vzdáleného parametru lze použít také bloky [GETPX](#) a [INCONN](#).

Tento blok propaguje kvalitu signálu. Více informací je uvedeno v sekci [1.4](#).

Vstup

GET	Vstup pro jednorázové přečtení parametru	Bool

Parametr

sc	Jméno vzdáleného parametru	String
GETF	Načtení parametru pouze po vyžádání	Bool
	off ... režim průběžného čtení parametru	
	on režim jednorázového přečtení parametru	
nmax	Rezervovaná paměť pro řetězec	Long (I32)

Výstup

sy	Hodnota parametru	String
E	Příznak chyby	Bool
	off ... bez chyby	
	on nastala chyba	

GETPX – Blok pro vzdálené získání parametru

Symbol bloku

Licence: [STANDARD](#)



Popis funkce

Blok **GETPX** funguje na stejném principu jako bloky **GETPB**, **GETPI**, **GETPR** a **GETPS**. Na rozdíl od těchto bloků je však univerzální a dokáže číst parametry všech typů kromě pole (array). Jméno vzdáleného parametru se zadává do parametru **sc** stejně jako v případě ostatních bloků. Typ hodnoty se nastavuje parametrem **type** a režim čtení parametru se nastavuje parametrem **GETF**. Pokud je parametr **GETF** nastaven na **on**, blok čte hodnotu parametru pouze po vyžádání na vstupu **GET**. Pokud je parametr **GETF** nastaven na **off**, blok čte hodnotu parametru průběžně.

Tento blok propaguje kvalitu signálu. Více informací je uvedeno v sekci [1.4](#).

Vstup

GET	Vstup pro jednorázové přečtení parametru	Bool

Parametr

sc	Jméno vzdáleného parametru	String
GETF	Načtení parametru pouze po vyžádání	Bool
	off ... režim průběžného čtení parametru	
	on režim jednorázového přečtení parametru	
type	Datový typ hodnoty	⊙8 Byte (U8)
	1 Bool	
	2 Byte (U8)	
	3 Short (I16)	
	4 Long (I32)	
	5 Word (U16)	
	6 DWord (U32)	
	7 Float (F32)	
	8 Double (F64)	
	9 Time	
	10 Large (I64)	
	11 Error	
	12 String	
	13 Reference	

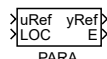
Výstup

y	Hodnota parametru	Any
E	Příznak chyby	Bool
	off ... bez chyby	
	on nastala chyba	

PARA – Blok s vektorovým parametrem nastavitelným ze vstupu

Symbol bloku

Licence: [STANDARD](#)



Popis funkce

Blok **PARA** je speciální blok, který kromě klasické metody zadávání svých parametrů umožňuje změnu jednoho svého parametru změnou vstupu. Parametr **apar** se může měnit podle vstupu **uRef**.

Logický vstup **LOC** (LOCal) určuje, zda bude hodnota vnitřního parametru **apar** čtena ze vstupu **uRef**, v tomto případě je **LOC = off**, nebo hodnota vnitřního parametru nebude na vstupu závislá (**LOC = on**). Pokud je blok v lokálním režimu **LOC = on**, je ve vnitřním parametru **apar** uložena poslední hodnota, která byla na vstupu **uRef** těsně před tím, než byl aktivován lokální režim (**LOC = off → on**).

Výstupní hodnota je shodná s hodnotou parametru **yRef = apar**.

Tento blok nepropaguje kvalitu signálu. Více informací je uvedeno v sekci [1.4](#).

Vstup

uRef	Odkaz na pole (vektor nebo matice)	Reference
LOC	Aktivace lokálního režimu	Bool
	off ... parametr je ovládán vstupním signálem	
	on lokální režim aktivován	

Parametr

SETS	Nastavení velikosti pole		Bool
nmax	Rezervovaná paměť pro pole	↓10 ⊙100	Long (I32)
etype	Typ prvků	⊙8	Long (I32)
	1 Bool		
	2 Byte (U8)		
	3 Short (I16)		
	4 Long (I32)		
	5 Word (U16)		
	6 DWord (U32)		
	7 Float (F32)		
	8 Double (F64)		
	10 Large (I64)		
apar	Interní hodnota parametru	⊙[0.0 1.0 2.0 3.0 4.0 5.0]	Double (F64)

Výstup

yRef

Odkaz na pole (vektor nebo matice)

Reference

PARE – Blok s parametrem výběr ze seznamu nastavitelným ze vstupu

Symbol bloku

Licence: [STANDARD](#)



Popis funkce

Blok je podobný bloku [PARI](#) s možností přiřadit texty číselným hodnotám. Odpovídající text je nastaven na výstupu **sy**. Blok má dva režimy a aktivní režim je určen parametrem **LIST**. Pokud je **LIST=off**, pak je na výstup **sy** zapsán odpovídající text. Pokud je **LIST=on**, pak je vstupní číslo považováno za bitové pole, texty jsou definovány pro každý bit a výstup **sy** je složen z textů, které odpovídají nastaveným bitům. Chování pro neznámé hodnoty určuje parametr **SATF**. Pokud je **SATF=off**, neznámá hodnota se nastaví na výstup **iy** a výstup **sy** je prázdný text. Při **SATF=on** jsou neznámé hodnoty ignorovány. Parametr **pupstr** má stejný formát jako v bloku **CNE**: **<number1>: <description1>|<number2>: <description2>|<number3>: <description3> ...**

Tento blok nepropaguje kvalitu signálu. Více informací je uvedeno v sekci [1.4](#).

Vstup

ip	Hodnota parametru	Long (I32)
LOC	Aktivace lokálního režimu	Bool
	off ... parametr je ovládán vstupním signálem	
	on lokální režim aktivován	

Parametr

ipar	Interní hodnota parametru	⊙1	Long (I32)
pupstr	Definice seznamu konstant		String
	⊙1: option A 2: option B 3: option C		
NUM	Číslo ve výstupním textu		Bool
LIST	Režim bitového pole		Bool
SATF	Saturace		Bool

Výstup

iy	Celočíselný výstupní signál	Long (I32)
sy	Výstupní textová hodnota	String

PARB, PARI, PARR – Bloky s nastavitelným parametrem ze vstupu

Symboly bloků

Licence: [STANDARD](#)



Popis funkce

Bloky PARR, PARI a PARB jsou speciální bloky, které kromě klasické metody zadávání svých parametrů umožňují změnu jednoho svého parametru změnou vstupu. U bloku:

- PARR změnu parametru **par** změnou vstupu **p**,
- PARI změnu **ipar** vstupem **ip**,
- PARB změnu **PAR** vstupem **P**.

Logický vstup **LOC** (LOCal) určuje, zda bude hodnota vnitřního parametru **par** (nebo **ipar**, **PAR**) čtena ze vstupu **p** (nebo **ip**, **P**), v tomto případě je **LOC = off**, nebo hodnota vnitřního parametru nebude na vstupu závislá (**LOC = on**). Pokud je blok v lokálním režimu **LOC = on**, je ve vnitřním parametru **par** (nebo **ipar**, **PAR**) uložena poslední hodnota, která byla na vstupu **p** (nebo **ip**, **P**) těsně před tím, než byl aktivován lokální režim (**LOC = off** → **on**). Následně je možno tuto hodnotu modifikovat ručně.

Výstupní hodnota je shodná s hodnotou parametru **y = par**, (nebo **k = ipar**, **Y = PAR**). Bloky PARR a PARI mají navíc možnost omezení výstupního signálu **y** a **k** saturačními mezemi **(lolim, hilim)**. Saturační omezení je uvažováno pouze v případě **SATF = on**.

Poznámka: Na stejném principu funguje blok [PARX](#), který umožňuje zadávat jakýkoliv typ parametru. Zvažte také použití bloku [SHLD](#), který lze rovněž použít pro ukládání číselné hodnoty, podobně jako u bloku PARR.

Tento blok nepropaguje kvalitu signálu. Více informací je uvedeno v sekci [1.4](#).

Vstup

p	Hodnota parametru	Double (F64)
LOC	Aktivace lokálního režimu	Bool
	off ... parametr je ovládán vstupním signálem	
	on lokální režim aktivován	

Parametr

par	Interní hodnota parametru	©1.0 Double (F64)

SATF	Saturace	Bool
	off ... signál není omezen	
	on saturační meze jsou aktivní	
hilim	Horní saturační mez	⊙1.0 Double (F64)
lolim	Dolní saturační mez	⊙-1.0 Double (F64)

Výstup

y	Analogový výstupní signál	Double (F64)

PARS – Blok s parametrem typu string nastavitelným ze vstupu

Symbol bloku

Licence: [STANDARD](#)



Popis funkce

Blok **PARS** má stejnou funkci jako bloky [PARR](#), [PARI](#) a [PARB](#), liší se pouze v tom, že nastavovaný parametr **spar** je typu string (řetězec) a je nastaven změnou vstupu **sp**.

Poznámka: Ke změně řetězcové hodnoty vzdáleného parametru lze použít také bloky [PARX](#) a [INCONN](#).

Tento blok nepropaguje kvalitu signálu. Více informací je uvedeno v sekci [1.4](#).

Vstup

<code>sp</code>	Hodnota parametru	String
<code>LOC</code>	Aktivace lokálního režimu	Bool

Parametr

<code>spar</code>	Interní hodnota parametru	String
<code>nmax</code>	Rezervovaná paměť pro řetězec	Long (I32)

Výstup

<code>sy</code>	Výstupní řetězec	String

PARX – Blok s parametrem nastavitelným ze vstupu

Symbol bloku

Licence: [STANDARD](#)



Popis funkce

Blok **PARX** stejně jako bloky **PARB**, **PARI**, **PARR** a **PARS** umožňuje změnu jednoho svého parametru změnou vstupu. Na rozdíl od výše zmíněných bloků může být parametr bloku **PARX** libovolného typu. Typ parametru se nastavuje parametrem **type**. U relevantních typů je možné nastavit saturační meze parametru.

Tento blok nepropaguje kvalitu signálu. Více informací je uvedeno v sekci [1.4](#).

Vstup

p	Hodnota parametru	Any
LOC	Aktivace lokálního režimu	Bool
	off ... parametr je ovládán vstupním signálem	
	on lokální režim aktivován	

Parametr

par	Interní hodnota parametru	⊙1.0	Double (F64)
SATF	Saturace		Bool
	off ... signál není omezen		
	on saturační meze jsou aktivní		
hilim	Horní saturační mez	⊙1.0	Double (F64)
lolim	Dolní saturační mez	⊙-1.0	Double (F64)
type	Datový typ hodnoty	⊙8	Byte (U8)
	1 Bool		
	2 Byte (U8)		
	3 Short (I16)		
	4 Long (I32)		
	5 Word (U16)		
	6 DWord (U32)		
	7 Float (F32)		
	8 Double (F64)		
	9 Time		
	10 Large (I64)		
	11 Error		
	12 String		
	13 Reference		

Výstup

y

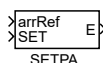
Hodnota parametru

Any

SETPA – Blok pro vzdálené nastavování vektorového parametru

Symbol bloku

Licence: [STANDARD](#)



Popis funkce

Blok SETPA slouží ke vzdálenému nastavování vektorových parametrů ostatních bloků v modelu. Může pracovat ve dvou režimech, které se přepínají parametrem SETF. Pro SETF = off je hodnota vzdáleného parametru **sc** nastavena na hodnotu vstupního vektoru **arrRef** při startu a dále pak při každé změně vstupního signálu. Jestliže parametr SETF je on, pak blok pracuje v režimu jednorázového zápisu vzdáleného parametru, který se nastaví vždy, když nastane náběžná hrana (off→on) na vstupu SET.

Jméno vzdáleného parametru určuje textový parametr **sc** (string connection), který se zadává ve tvaru <cesta_k_bloku:jmeno_parametru>. Cesta k bloku, jehož parametr má být získán, může obsahovat tečkami oddělené hierarchické úrovně, na jejichž konci je název bloku a může být:

- **Relativní:**

- Začíná znakem '.', tedy v úrovni, do které je umístěn blok SETPA. Příklady cest: ".CNDR:yp", ".Lights.ATMT:touts".
- Začíná znaky '..', tedy v úrovni nad blokem SETPA. Příklady: "..CNDR:yp", "..Lights.ATMT:touts".

- **Relativní k tasku:** Začíná v základní úrovni tasku, do které je umístěn daný blok SETPA. V tomto případě text začíná znakem '%'. Příklady hodnot cest: "%CNDR:yp", "%Lights.ATMT:touts".

- **Absolutní:** Úplná posloupnost hierarchických úrovní až k požadovanému bloku. V případě, že má být čten parametr z bloku umístěného v úloze ovladače (pro konfiguraci viz. blok [IOTASK](#)), je v první úrovni hierarchie uveden znak '&' následovaný názvem ovladače. Příklady hodnot absolutních cest: "uloha1.vstupy.ATMT:touts", "&EfaDrv.mereni.CNDR:yp".

Pořadí a názvy jednotlivých hierarchických úrovní jsou zobrazeny ve stromové struktuře konfigurace v diagnostice programu REXYGEN Studio.

Varování: Pokud se zapisuje hodnota do jiné úlohy, je pro zajištění konzistence hodnoty nutné použít tzv. semafor a počkat na dokončení úlohy, do které se hodnota zapisuje. Po celou dobu čekání na dokončení je úloha s blokem SETPA pozastavena! Z

praktického hlediska to znamená, že blok **SETPA** se musí umístit do úlohy, která trvá dlouho a zapisovat hodnotu do úlohy, která trvá krátce. Pokud je to opačně, dochází k čekání rychlé úlohy na pomalejší úlohu a rychlejší úloha se tím zpozdí. V této situaci je vhodné použít blok **GETPA** v déletrvající úloze.

Poznámka 1: Pokud se použije více bloků **SETPA** pro nastavení hodnot v jiném tasku, není zajištěno, že se všechny hodnoty nastaví v jedné periodě druhé úlohy. Je pouze zajištěno, že dříve provedený blok **SETPA** nastaví hodnotu ve stejné nebo dřívější periodě druhého tasku než později provedený blok **SETPA**. Pořadí spouštění je vidět v diagnostice programu REXYGEN Studio.

Poznámka 2: Vzdálené pole (parametr na který blok odkazuje) může být jak primární pole (např. **CNA:acn**, **RTOV:xVec**, **MX_MAT:ay**), tak reference na pole (např. **CNA:vec**, **RTOV:yVec**, **SUBSYSTEM:Outport**). Reference na pole by měla být umístěna ve stejném tasku, jako primární pole, jinak hrozí nekonzistence.

Poznámka 3: Pro vzdálené zapisování hodnoty lze použít i blok **OUTCONN**.

Tento blok propaguje kvalitu signálu. Více informací je uvedeno v sekci 1.4.

Vstup

arrRef	Odkaz na pole (vektor nebo matice)	Reference
SET	Vstup pro jednorázový zápis parametru	Bool

Parametr

sc	Jméno vzdáleného parametru	String
SETF	Nastavení parametru pouze na vyžádání	Bool
	off ... režim průběžného nastavování parametru	
	on režim jednorázového nastavení parametru	
SETS	Nastavení velikosti pole	Bool

Výstup

E	Příznak chyby	Bool
	off ... bez chyby	
	on nastala chyba	

SETPB, SETPI, SETPR – Bloky pro vzdálené nastavování parametru

Symbole bloků

Licence: [STANDARD](#)



Popis funkce

Bloky **SETPR**, **SETPI**, **SETPB** slouží pro vzdálené nastavování parametrů ostatních bloků v modelu. Bloky mají identickou funkci, liší se pouze v typu parametru, který nastavují. Blok **SETPR** je pro reálné číslo, **SETPI** pro celé číslo a **SETPB** pro Booleovskou hodnotu. Kvůli dodržení konvence v pojmenovávání proměnných [1.3](#) se vstupy a výstupy jednotlivých bloků nazývají podle typu nastavovaného parametru:

- **p**, **y** – reálný vstup a výstup bloku **SETPR**,
- **ip**, **k** – celočíselný vstup a výstup **SETPI**,
- **P**, **Y** – Boolean vstup a výstup pro **SETPB**.

Bloky mohou pracovat ve dvou režimech, které se přepínají parametrem **SETF**. Pro **SETF = off** je hodnota vzdáleného parametru **sc** nastavena na hodnotu vstupního parametru **p** (nebo **ip**, **P**) při startu a dále pak při každé změně vstupního parametru **p** (nebo **ip**, **P**). V případě **SETF = on** bloky pracují v režimu jednorázového zápisu vzdáleného parametru, který se zapíše při každé náběžné hraně (**off**→**on**) na vstupu **SET**. Po úspěšném zápisu je výstup **y** (nebo **k**, **Y**) nastaven na zapisovanou hodnotu a chybový výstup **E = off**. Při neúspěšném zápisu je **E = on**.

Jméno vzdáleného parametru určuje textový parametr **sc** (string connection), který se zadává ve tvaru `<cesta_k_bloku:jmeno_parametru>`. Rovněž je možné přistupovat k jednotlivým prvkům parametrů typu pole (např. parametr **tout** bloku [ATMT](#)). Toho se dosáhne pomocí hranatých závorek a čísla prvku, např. tedy `.ATMT:touts[2]`, číslování je od 0, uvedený propojovací řetězec tedy odkazuje na třetí prvek pole.

Cesta k bloku, jehož parametr má být nastavován, může obsahovat tečkami oddělené hierarchické úrovně, na jejichž konci je název bloku a může být:

- **Relativní:**

- Začíná znakem '.', tedy v úrovni, do které je umístěn blok **SETPR** (nebo **SETPI**, **SETPB**). Příklady cest: `".GAIN:k"`, `".Motor1.Poloha:ycn"`.
- Začíná znaky '..', tedy v úrovni nad blokem **SETPR** (nebo **SETPI**, **SETPB**). Příklady cest: `".GAIN:k"`, `".Motor1.Poloha:ycn"`.

- **Relativní k tasku:** Začíná v základní úrovni tasku, do které je umístěn daný blok SETPR (nebo SETPI, SETPB). V tomto případě text začíná znakem '%'. Příklady hodnot cest: "%GAIN:k", "%Motor1.Poloha:ycn".
- **Absolutní:** Úplná posloupnost hierarchických úrovní až k požadovanému bloku. V případě, že má být čten parametr z bloku umístěného v úloze ovladače (pro konfiguraci viz. blok IOTASK), je v první úrovni hierarchie uveden znak '&' následovaný názvem ovladače. Příklady hodnot absolutních cest: "uloha1.vstupy.lin1:u2", "&EfaDrv.mereni.DER1:n".

Pořadí a názvy jednotlivých hierarchických úrovní jsou zobrazeny ve stromové struktuře konfigurace v diagnostice programu REXYGEN Studio.

Varování: Pokud se zapisuje hodnota do jiné úlohy, je pro zajištění konzistence hodnoty nutné použít tzv. semafor a počkat na dokončení úlohy do které se hodnota zapisuje. Po celou dobu čekání na dokončení je úloha s blokem SETPx pozastavena! Z praktického hlediska to znamená, že blok SETPx se musí umístit do úlohy, která trvá dlouho a zapisovat hodnotu do úlohy, která trvá krátce. Pokud je to opačně, dochází k čekání rychlé úlohy na pomalejší úlohu a rychlejší úloha se tím zpozdí. V této situaci je vhodné použít blok GETPx v déletrvající úloze.

Poznámka 1: Pokud se použije více bloků SETPx pro nastavení hodnot v jiném tasku, není zajištěno, že se všechny hodnoty nastaví v jedné periodě druhé úlohy. Je pouze zajištěno, že dříve provedený blok SETPx nastaví hodnotu ve stejné nebo dřívější periodě druhého tasku než později provedený blok SETPx. Pořadí spouštění je vidět v diagnostice programu REXYGEN Studio. Pokud je důležité zapsat všechny hodnoty ve stejné periodě, musí se přenášet pomocí bloků Inport a Outport nebo z hodnot vytvořit pole a zapsat je najednou blokem SETPA.

Poznámka 2: Pro vzdálené zapisování hodnoty lze použít i bloky SETPX a OUTCONN.

Tento blok propaguje kvalitu signálu. Více informací je uvedeno v sekci 1.4.

Vstup

p	Požadovaná hodnota parametru	Double (F64)
SET	Vstup pro jednorázový zápis parametru	Bool

Parametr

sc	Jméno vzdáleného parametru	String
SETF	Nastavení parametru pouze na vyžádání	Bool
	off ... režim průběžného nastavování parametru	
	on režim jednorázového nastavení parametru	

Výstup

y	Hodnota parametru	Double (F64)

E	Příznak chyby	Bool
	off ... bez chyby	
	on nastala chyba	

SETPS – Blok pro vzdálené nastavování parametru typu string

Symbol bloku

Licence: [STANDARD](#)



Popis funkce

Blok **SETPS** má stejnou funkci jako bloky [SETPR](#), [SETPI](#) a [SETPB](#), liší se pouze v tom, že nastavuje hodnotu parametru typu string (řetězec).

Poznámka: K zápisu hodnoty vzdáleného parametru lze použít také bloky [SETPX](#) a [OUTCONN](#).

Tento blok propaguje kvalitu signálu. Více informací je uvedeno v sekci [1.4](#).

Vstup

<code>sp</code>	Požadovaná hodnota parametru	String
<code>SET</code>	Vstup pro jednorázový zápis parametru	Bool

Parametr

<code>sc</code>	Jméno vzdáleného parametru	String
<code>SETF</code>	Nastavení parametru pouze na vyžádání	Bool
<code>nmax</code>	Rezervovaná paměť pro řetězec	Long (I32)

Výstup

<code>sy</code>	Hodnota parametru	String
<code>E</code>	Příznak chyby	Bool

SETPX – Blok pro vzdálené nastavování parametru

Symbol bloku

Licence: [STANDARD](#)



Popis funkce

Blok **SETPX** funguje na stejném principu jako bloky [SETPB](#), [SETPR](#), [SETPS](#) a [SETPI](#). Na rozdíl od těchto bloků je však univerzální a dokáže nastavit parametry všech typů kromě pole (array). Jméno vzdáleného parametru se zadává do parametru **sc** stejně jako v případě ostatních bloků. Typ hodnoty se nastavuje parametrem **type** a režim zápisu parametrem **SETF**. Pokud je parametr **SETF** nastaven na **on**, blok nastaví hodnotu parametru pouze po vyžádání na vstupu **SET**. Pokud je parametr **SETF** nastaven na **off**, blok nastaví hodnotu parametru průběžně.

Tento blok propaguje kvalitu signálu. Více informací je uvedeno v sekci [1.4](#).

Vstup

p	Požadovaná hodnota parametru	Any
SET	Vstup pro jednorázový zápis parametru	Bool

Parametr

sc	Jméno vzdáleného parametru	String
SETF	Nastavení parametru pouze na vyžádání	Bool
	off ... režim průběžného nastavování parametru on ... režim jednorázového nastavení parametru	
type	Datový typ hodnoty	⊙8 Byte (U8)
	1 Bool	
	2 Byte (U8)	
	3 Short (I16)	
	4 Long (I32)	
	5 Word (U16)	
	6 DWord (U32)	
	7 Float (F32)	
	8 Double (F64)	
	9 Time	
	10 Large (I64)	
	11 Error	
	12 String	
	13 Reference	

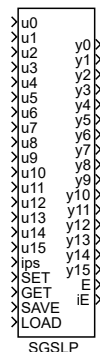
Výstup

y	Hodnota parametru	Any
E	Příznak chyby	Bool
	off ... bez chyby	
	on nastala chyba	

SGSLP – Nastavování, čtení, ukládání a načítání parametrů

Symbol bloku

Licence: [ADVANCED](#)



Popis funkce

Blok **SGSLP** (z anglického *Set, Get, Save and Load Parameters*) je speciálním blokem pro správu připojených parametrů jiných bloků v konfiguraci řídicího systému **REXYGEN**. Blok pracuje i v systému Matlab-Simulink, jeho dosah je však omezen jen na bloky téhož souboru **.mdl**, v němž je vložen.

Blok může pracovat až se šestnácti sadami parametrů, které jsou číslovány od 0 do 15 a volí se vstupem **ips**, aktuální počet sad je určen parametrem **nps**. Je-li vstup **ips** nepřipojen, pracuje blok se sadou **ips = 0**. V každé sadě může být zkonfigurováno až 16 různých parametrů daných řetězcovými parametry **sc0** až **sc15**, takže jeden blok **SGSLP** může pracovat s maximálně 256 parametry v řídicím systému **REXYGEN**. Je-li řetězec **sci** prázdný (nezadaný), není žádný parametr specifikován, jinak parametry **sci** mohou používat dvě syntaxe:

1. **<blok>:<param>** – specifikují jeden blok se jménem **blok** s parametrem **param**. V tomto případě je použit tentýž blok a parametr pro všech **nps** sad parametrů.
2. **<blok>:<param><sep>...<blok>:<param>** – v tomto případě je pro každou sadu parametrů **ips** uvažován obecně různý parametr, první dvojice **<blok>:<param>** odpovídá **ips = 0**. Oddělovačem **<sep>** může být buď čárka nebo středník. Specifikovaných dvojic **<blok>:<param>** by mělo být právě **nps**. V případě, že jich je méně, a má se provést některá z operací (viz níže) se sadou, pro niž specifikace bloku a parametru chybí, požadovaná operace se neprovede.

Doporučuje se nepoužívat obě syntaxe v jednom **SGSLP** bloku, všech 16 řetězců **sci** by mělo mít stejný formát. První syntaxe se například používá při výrobě **nps** typů výrobků, kdy je třeba měnit mnoho parametrů pro každý typ výroby. Druhá syntaxe se obvykle používá pro ukládání uživatelsky definovaných parametrů na disk (viz operace **SAVE** níže).

V tomto případě je žádoucí zajistit automatizované přepínání vstupu **ips** (např. pomocí bloku [ATMT](#) z knihovny **LOGIC**).

Pokud všechny bloky, jejichž parametry mají být nastavovány daným blokem **SGSLP**, leží v hierarchii bloků v nějakém subsystému nebo níže, lze výhodně použít řetězcový parametr **broot**, v němž se uvede jméno tohoto subsystému. Toto jméno se připojuje před každou specifikaci **<blok>** v parametrech **sci**. V případě, že je **broot = '.'**, je výsledek stejný, jako by parametr obsahoval cestu k subsystému, do nějž je daný blok **SGSLP** vložen (parametr se zadává bez uvozovek, ty jsou použity pouze v tomto textu pro zvýraznění jednotlivého znaku). Je-li hodnota parametru **broot** prázdná, musí každý výskyt **<blok>** v parametrech **sci** specifikovat úplnou cestu k bloku, v níž jsou jednotlivé hierarchické úrovně odděleny tečkami. Například tedy volba **broot = .** a **sc0 = CNR:ycn** zajišťuje propojení na blok **CNR** a jeho parametr **ycn**, který se nachází ve stejném subsystému jako blok **SGSLP**. Případně můžeme ponechat parametr **broot** prázdný a umístit znak **'.'** na začátek řetězce **sc0**. Blížší informace o cestách v systému **REXYGEN** jsou uvedeny u bloků [GETPR](#) a [SETPR](#).

Blok **SGSLP** může při náběžné hraně (**off**→**on**) na některém ze stejnojmenných vstupů provádět následující operace:

SET – nastavit parametry dané množiny **ips** na hodnoty přivedené na vstupy **ui**. V případě, že je parametr úspěšně nastaven, je na stejnou hodnotu nastaven i výstup **yi**.

GET – získat parametry dané množiny **ips**. V případě, že je parametr úspěšně získán, je jeho hodnota nastavena na výstup **yi**.

SAVE – uložit parametry dané množiny **ips** do souboru (tzv. stavový soubor) na cílovém zařízení. Parametry a formát souboru jsou popsány níže.

LOAD – načíst parametry dané množiny **ips** ze souboru na cílovém zařízení. Kromě načtení parametrů při náběžné hraně vstupu **LOAD** se parametry sady **ips0** načtou při inicializaci bloku v případě, že je hodnota parametru **ips0** v rozsahu od 0 do **nps** – 1. Parametry a formát souboru jsou popsány níže.

Operace **LOAD** a **SAVE** pracují se souborem na cílovém zařízení, jehož jméno je uvedeno v parametru **fname**. Práce s parametrem **fname** se řídí následujícími pravidly:

- Pokud jméno souboru neobsahuje příponu, přidává se automaticky přípona **.rxs** (ReX Status file).
- Při ukládání bude vytvářen záložní soubor se stejným jménem, avšak s příponou modifikovanou přidáním znaku **'~'** ihned za znak **'.'**, např. pokud jméno souboru neobsahuje příponu, je přípona záložního souboru **~rxs**.
- Cesta je relativní a je vztažena k adresáři s datovými soubory runtime jádra systému **REXYGEN** na cílovém zařízení. Data se typicky ukládají na pevný disk nebo flash disk nebo jiné médium, které po vypnutí a opětovném zapnutí zachovává soubory.

Data jsou příkazem **SAVE** ukládána do textového souboru, ze kterého jsou příkazem **LOAD** načítána zpět do bloku **SGSLP**. Pro každý parametr **sci**, $i = 0, \dots, m$, kde $m < 16$ je maximální číslo, pro něj je parametr **scm** neprázdný řetězec, obsahuje soubor dva řádky ve tvaru:

```
"<blok>:<param>", ..., "<blok>:<param>"

<hodnota>, ..., <hodnota>
```

Jednotlivé položky "<blok>:<param>" jsou mezi sebou odděleny čárkami a jejich počet odpovídá parametru **nps**, obdobně to platí i o položkách <hodnota> obsahujících hodnotu parametru, jehož jméno je uvedeno ve stejné pozici v předchozím řádku. Poznamenejme, že pro **nps** > 1 má první z těchto dvou řádků vždy právě uvedený tvar (dvojice "<blok>:<param>" se opakuje **nps**-krát) a to i v případě, že parametr **sci** obsahuje jedinou dvojici <blok>:<param> (viz 1. syntaxe výše). Tato skutečnost umožňuje přecházet mezi oběma syntaxemi parametrů **sci**, aniž by musel být soubor upravován. Při ukládání malého počtu hodnot můžete rovněž využít blok [SILO](#).

Tento blok nepropaguje kvalitu signálu. Více informací je uvedeno v sekci [1.4](#).

Vstup

u0..u15	Analogový vstupní signál	Double (F64)
ips	Číslo sady parametrů (číslováno od 0)	Long (I32)
SET	Přečtení vstupů a nastavení parametrů	Bool
GET	Přečtení parametrů a nastavení výstupů	Bool
SAVE	Uložení parametrů do souboru	Bool
LOAD	Načtení parametrů ze souboru	Bool

Parametr

nps	Počet sad parametrů	↓1 ↑16 ⊙1	Long (I32)
ips0	Číslo sady parametrů pro inicializaci bloku	↓-1 ↑15	Long (I32)
iprec	Počet platných číslic pro hodnoty typu double	↓2 ↑15 ⊙12	Long (I32)
icolw	Šířka sloupce v souboru	↓0 ↑22	Long (I32)
fname	Jméno souboru pro ukládání/načítání parametrů	⊙status	String
broot	Kořenový prvek v hierarchii	⊙.	String
sc0..sc15	Seznam přiřazených parametrů		String

Výstup

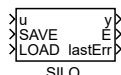
y0..y15	Analogový výstupní signál	Double (F64)
E	Příznak chyby	Bool
	off ... bez chyby	
	on nastala chyba	

iE	Chybový nebo varovný výstup	Long (I32)
	0 Operace proběhla úspěšně	
	1 Fatální chyba systému Matlab	
	2 Chyba při operaci LOAD	
	3 Chyba při operaci SAVE	
	4 Nesprávný formát souboru	
	5 Sada parametrů ips nebyla v souboru nalezena	
	6 Nesoulad jmen parametrů	
	7 Neočekávaný konec souboru	
	8 Chyba zápisu na disk (nedostatek místa?)	
	9 Chyba syntaxe parametru	
	10 Jméno tvořeno pouze mezerami	
	11 Nelze vytvořit záložní soubor	
	12 Chyba při operaci GET	
	13 Chyba při operaci SET	
	14 Překročení času (timeout)	
	15 Parametr je pouze pro čtení	
	16 Sada číslo ips neexistuje	

SILO – Uložení vstupního signálu, načtení výstupního signálu

Symbol bloku

Licence: [STANDARD](#)



Popis funkce

Blok **SILO** je určen pro export nebo import jednoho signálu (hodnoty) do nebo ze souboru. Hodnota je uložena při náběžné hraně (**off**→**on**) na vstupu **SAVE** a po úspěšném uložení je nastavena také na výstup **y**. Načtení hodnoty probíhá při startu a při náběžné hraně (**off**→**on**) na vstupu **LOAD**.

Chyba diskové operace je indikována na výstupech **E** a **lastErr**. Příznak **E** je shozen při sestupné hraně na vstupu **SAVE** nebo **LOAD**, zatímco výstup **lastErr** drží hodnotu až do další operace. Pokud chyba nastala při operaci **LOAD**, je na výstup **y** nastavena náhradní hodnota **yerr**.

Alternativně lze zapnout průběžné ukládání nebo čtení pomocí příslušného parametru (**CSF**, **CLF**). Diskové operace pak probíhají kontinuálně, ovšem pouze když je příslušný vstupní signál nastaven na **on**. Pozor však na to, že zápis/čtení pak probíhá při každém spuštění bloku, což může mít za následek nadměrné zatížení úložného zařízení, proto je potřeba použití tohoto režimu vždy důkladně zvážit.

Parametr **fname** určuje umístění souboru. Cesta je relativní a je vztažena k adresáři s datovými soubory runtime jádra systému **REXYGEN** na cílovém zařízení.

Pro pokročilé a hromadné operace je určen blok [SGSLP](#).

Tento blok nepropaguje kvalitu signálu. Více informací je uvedeno v sekci [1.4](#).

Vstup

u	Vstupní signál	Double (F64)
SAVE	Uložení vstupní hodnoty do souboru	Bool
LOAD	Načtení hodnoty výstupu ze souboru	Bool

Parametr

fname	Jméno souboru pro ukládání/načítání parametrů	String
CSF	Průběžné ukládání	Bool
CLF	Průběžné načítání	Bool
yerr	Náhradní hodnota pro případ chyby	Double (F64)

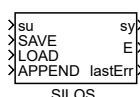
Výstup

y	Výstupní signál	Double (F64)
E	Příznak chyby	Bool
	off ... bez chyby	
	on nastala chyba	
lastErr	Výsledek poslední operace	Long (I32)

SILOS – Uložení vstupního řetězce, načtení výstupního řetězce

Symbol bloku

Licence: [STANDARD](#)



Popis funkce

Blok **SILOS** je určen pro export nebo import jednoho řetězce do nebo ze souboru. Řetězec je uložen při náběžné hraně (**off**→**on**) na vstupu **SAVE** a po úspěšném uložení je nastavena také na výstup **sy**. Načtení hodnoty probíhá při startu a při náběžné hraně (**off**→**on**) na vstupu **LOAD**.

Pokud je na vstup **APPEND** přivedena hodnota **on**, řetězec ze vstupu je při ukládání přidán na konec souboru. Tento režim se hodí pro logování událostí do textových souborů. Na načítání ze souboru nemá tento vstup žádný vliv.

Pomocí parametru **LLO** lze zvolit, zda se má načítat celý soubor (**off**) nebo pouze jeho poslední řádek (**on**).

Chyba diskové operace je indikována na výstupech **E** a **lastErr**. Příznak **E** je shozen při sestupné hraně na vstupu **SAVE** nebo **LOAD**, zatímco výstup **lastErr** drží hodnotu až do další operace.

Alternativně lze zapnout průběžné ukládání nebo čtení pomocí příslušného parametru (**CSF**, **CLF**). Diskové operace pak probíhají kontinuálně, ovšem pouze když je příslušný vstupní signál nastaven na **on**. Pozor však na to, že zápis/čtení pak probíhá při každém spuštění bloku, což může mít za následek nadměrné zatížení úložného zařízení, proto je potřeba použití tohoto režimu vždy důkladně zvážit.

Parametr **fname** určuje umístění souboru. Cesta je relativní a je vztažena k adresáři s datovými soubory runtime jádra systému **REXYGEN** na cílovém zařízení.

Tento blok nepropaguje kvalitu signálu. Více informací je uvedeno v sekci [1.4](#).

Vstup

su	Vstupní řetězec	⊙0	String
SAVE	Uložení vstupního řetězce do souboru		Bool
LOAD	Načtení řetězce ze souboru		Bool
APPEND	Připojení ukládaného řetězce k souboru		Bool

Parametr

<code>fname</code>	Jméno souboru pro ukládání/načítání parametrů	String
<code>CSF</code>	Průběžné ukládání	Bool
<code>CLF</code>	Průběžné načítání	Bool
<code>LL0</code>	Načítání jen poslední řádky	Bool
<code>nmax</code>	Rezervovaná paměť pro řetězec	↓0 ↑65520 Long (I32)

Výstup

<code>sy</code>	Výstupní řetězec	String
<code>E</code>	Příznak chyby <code>off ...</code> bez chyby <code>on</code> nastala chyba	Bool
<code>lastErr</code>	Výsledek poslední operace	Long (I32)

Kapitola 13

MODEL – Simulace dynamických systémů

Obsah

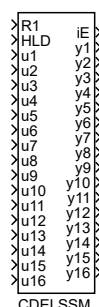
CDELSSM – Stavový model spojitého lineárního systému s dopravním zpožděním	417
CSSM – Stavový model spojitého lineárního systému	420
DDELSSM – Stavový model diskrétního lineárního systému s dopravním zpožděním	423
DFIR – Filtr s konečnou impulzní odezvou	425
DSSM – Stavový model diskrétního lineárního systému	426
EKF – Rozšířený (nelineární) Kalmanův filtr	428
FOPDT – Model systému 1. řádu s dopravním zpožděním	431
IPEN2, IPEN3 – N-ramenné inverzní kyvadlo na vozíku - Fyzikální parametry	432
IPEN2pu, IPEN3pu – N-ramenné inverzní kyvadlo na vozíku - Dynamické parametry	435
MDL – Model procesu	437
MDLI – Model procesu s proměnnými parametry	438
MVD – Motorizovaný pohon ventilu	439
NSSM – Nelineární stavový model	440
NUREACT – Model of nuclear reactor	443
QCOPT – Model dronu (kvadrukoptéry)	444
SGEN – Model synchronního generátoru	446
SGENTX – Model synchronního generátoru	448
SOPDT – Model systému 2. řádu s dopravním zpožděním	450
STMGEN – Model parogenerátoru	452
STURB – Model parní turbíny	454

Knihovna MODEL je zaměřena na modelování a simulaci systému. Zahrnuje bloky jako **CSSM** a **DSSM** pro spojité a diskrétní modely stavového prostoru a **DFIR** pro digitální filtry s konečnou impulsní odezvou. Knihovna nabízí **EKF** pro implementaci rozšířeného Kalmanova filtru a **FOPDT**, **SOPDT** pro modely časového zpoždění procesu prvního a druhého řádu. Navíc poskytuje **FMUCS** a **FMUINFO** pro propojení s funkčními mock-upovými jednotkami a **MDL**, **MDLI** pro rozhraní generických modelů. Pokročilé funkce jsou pokryty bloky jako **CDELSSM**, **DDELSSM** pro stavové modely kontinuálních a diskrétních systémů se zpožděním a **MVD** pro modelové proměnné zpoždění, které splňují širokou škálu požadavků na modelování v systému REXYGEN.

CDELSSM – Stavový model spojitého lineárního systému s dopravním zpožděním

Symbol bloku

Licence: [ADVANCED](#)



Popis funkce

Funkční blok CDELSSM (Continuous State Space Model with time DELay) simuluje chování lineárního spojitého systému s dopravním zpožděním *del* ve stavové reprezentaci:

$$\begin{aligned}\frac{dx(t)}{dt} &= A_c x(t) + B_c u(t - del), \quad x(0) = x_0 \\ y(t) &= C_c x(t) + D_c u(t),\end{aligned}$$

kde $x(t) \in \mathbb{R}^n$ je vektor stavu, $x_0 \in \mathbb{R}^n$ je počáteční hodnota vektoru stavu, $u(t) \in \mathbb{R}^m$ je vektor vstupu, $y(t) \in \mathbb{R}^p$ je vektor výstupu. Matice $A_c \in \mathbb{R}^{n \times n}$ určuje dynamiku systému, matice $B_c \in \mathbb{R}^{n \times m}$ určuje působení vstupu na stav systému, matice $C_c \in \mathbb{R}^{p \times n}$ určuje působení stavu na výstup systému a matice $D_c \in \mathbb{R}^{p \times m}$ určuje přímé působení vstupu na výstup systému. Pokud je **UD=off**, matice D_c se při simulaci nepoužívá (chová se jako by byla nulová).

Všechny matice se zadávají stejným způsobem jako v systému Matlab, tj. celá matice je uzavřena v hranatých závorkách, zadává se po řádcích, jednotlivé prvky v řádku se oddělují mezerou, jednotlivé řádky středníkem. Pro oddělení desetinné části čísla se používá tečka. Vektor x_0 je sloupcový, proto se všechny jeho prvky oddělují středníkem (každý prvek je na samostatném řádku).

Simulovaný systém se nejprve převede do diskrétního (diskretizovaného) stavového modelu:

$$\begin{aligned}x((k+1)T) &= A_d x(kT) + B_{d1} u((k-d)T) + B_{d2} u((k-d+1)T), \quad x(0) = x_0 \\ y(kT) &= C_c x(kT) + D_c u(kT),\end{aligned}$$

kde $k \in \{1, 2, \dots\}$ je krok simulace, T je perioda spouštění bloku v [s] a d je zpoždění v krocích simulace tak, aby $(d-1)T < del \leq d.T$. Perioda T se v bloku nezadává,

je určena automaticky jako perioda úlohy (**TASK**, **QTASK** nebo **IOTASK**), do níž je blok zařazen.

Vstupy simulovaného systému **u1..u16** reprezentují vektor vstupu **u(t)**. Pro danou simulaci se používá prvních m vstupů, kde m je počet sloupců matice **Bc**. Pokud se vstup $u(t)$ mění jen v okamžicích vzorkování a mezi dvěma sousedními vzorkovacími okamžiky je konstantní (což se předpokládá), tj. $u(t) = u(kT)$ pro $t \in [kT, (k+1)T)$, pak matice A_d , B_{d1} a B_{d2} jsou určeny vztahy:

$$\begin{aligned} A_d &= e^{A_c T} \\ B_{d1} &= e^{A_c(T-\Delta)} \int_0^\Delta e^{A_c \tau} B_c d\tau \\ B_{d2} &= \int_0^{T-\Delta} e^{A_c \tau} B_c d\tau, \end{aligned}$$

kde $\Delta = del - (d-1)T$.

Výpočet diskretních matic A_d , B_{d1} a B_{d2} je založen na metodě popsané v [8], využívající Padéových aproximací maticové exponenciály a jejího integrálu a měřítkování.

Při simulaci v reálném čase se pak v každém okamžiku spuštění bloku vždy vypočte jeden krok podle diskretního stavového modelu uvedeného výše. Výstupy simulovaného systému **y1..y16** reprezentují stav systému **x(t)** a pro danou simulaci se používá prvních p výstupů, kde p je počet řádků matice **Cc**.

Výstup **iE** je celočíselný a obsahuje informaci o průběhu simulace:

- **0**: vše v pořádku, blok simuluje správně
- **-213**: nekompatibilita rozměrů matic stavového modelu
- **-510**: úloha je špatně podmíněná (některá z pracovních matic je singulární nebo blízká singulární matici)
- **xxx**: chybový kód **xxx** systému REXYGEN, více viz přílohu C

Tento blok propaguje kvalitu signálu. Více informací je uvedeno v sekci 1.4.

Vstup

R1	Reset bloku	Bool
HLD	Podržení aktuálního stavu modelu	Bool
u1..u16	Analogový vstupní signál	Double (F64)

Parametr

UD	Příznak použití matice Dc	Bool
del	Dopravní zpoždění modelu [s]	↓0.0 Double (F64)
is	Řád Padéovy aproximace	↓0 ↑4 ⊙2 Long (I32)
eps	Přesnost aproximace	↓0.0 ↑1.0 ⊙1e-15 Double (F64)

Ac	Matice A spojitého modelu	$\odot [-0.36 \ -1.24 \ -0.18; \ 1 \ 0 \ 0; \ 0 \ 1 \ 0]$	Double (F64)
Bc	Matice B spojitého modelu	$\odot [0.5; \ 0; \ 0]$	Double (F64)
Cc	Matice C spojitého modelu	$\odot [0.12 \ 0.48 \ 0.36]$	Double (F64)
Dc	Matice D spojitého modelu	$\odot [0]$	Double (F64)
x0	Počáteční hodnota stavu x	$\odot [0; \ 0; \ 0]$	Double (F64)

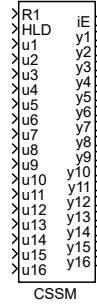
Výstup

iE	Kód chyby	Error
y1..y16	Analogový výstupní signál	Double (F64)

CSSM – Stavový model spojitého lineárního systému

Symbol bloku

Licence: [ADVANCED](#)



Popis funkce

Funkční blok **CSSM** (Continuous State Space Model) simuluje chování lineárního spojitého systému ve stavové reprezentaci:

$$\begin{aligned}\frac{dx(t)}{dt} &= A_c x(t) + B_c u(t), \quad x(0) = x_0 \\ y(t) &= C_c x(t) + D_c u(t),\end{aligned}$$

kde $x(t) \in \mathbb{R}^n$ je vektor stavu, $x_0 \in \mathbb{R}^n$ je počáteční hodnota vektoru stavu, $u(t) \in \mathbb{R}^m$ je vektor vstupu, $y(t) \in \mathbb{R}^p$ je vektor výstupu. Matice $A_c \in \mathbb{R}^{n \times n}$ určuje dynamiku systému, matice $B_c \in \mathbb{R}^{n \times m}$ určuje působení vstupu na stav systému, matice $C_c \in \mathbb{R}^{p \times n}$ určuje působení stavu na výstup systému a matice $D_c \in \mathbb{R}^{p \times m}$ určuje přímé působení vstupu na výstup systému. Pokud je **UD=off**, matice D_c se při simulaci nepoužívá (chová se jako by byla nulová).

Všechny matice se zadávají stejným způsobem jako v systému Matlab, tj. celá matice je uzavřena v hranatých závorkách, zadává se po řádcích, jednotlivé prvky v řádku se oddělují mezerou, jednotlivé řádky středníkem. Pro oddělení desetinné části čísla se používá tečka. Vektor x_0 je sloupcový, proto se všechny jeho prvky oddělují středníkem (každý prvek je na samostatném řádku).

Simulovaný systém se nejprve převede do diskrétního (diskretizovaného) stavového modelu:

$$\begin{aligned}x((k+1)T) &= A_d x(kT) + B_d u(kT), \quad x(0) = x_0 \\ y(kT) &= C_c x(kT) + D_c u(kT),\end{aligned}$$

kde $k \in \{1, 2, \dots\}$ je krok simulace, T je perioda spouštění bloku v [s]. Perioda T se v bloku nezadává, je určena automaticky jako perioda úlohy (**TASK**, **QTASK** nebo **IOTASK**), do níž je blok zařazen.

Vstupy simulovaného systému $u1..u16$ reprezentují vektor vstupu $u(t)$. Pro danou simulaci se používá prvních m vstupů, kde m je počet sloupců matice B_c . Pokud se vstup $u(t)$ mění jen v okamžicích vzorkování a mezi dvěma sousedními vzorkovacími okamžiky je konstantní (což se předpokládá), tj. $u(t) = u(kT)$ pro $t \in [kT, (k+1)T)$, pak matice A_d a B_d jsou určeny vztahy

$$\begin{aligned} A_d &= e^{A_c T} \\ B_d &= \int_0^T e^{A_c \tau} B_c d\tau \end{aligned}$$

Výpočet diskretních matic A_d a B_d je založen na metodě popsané v [8], využívající Padéových aproximací maticové exponenciály a jejího integrálu a měřítkování.

Při simulaci v reálném čase se pak v každém okamžiku spuštění bloku vždy vypočte jeden krok podle diskretního stavového modelu uvedeného výše. Výstupy simulovaného systému $y1..y16$ reprezentují stav systému $x(t)$ a pro danou simulaci se používá prvních p výstupů, kde p je počet řádků matice C_c .

Výstup **iE** je celočíselný a obsahuje informaci o průběhu simulace:

- **0**: vše v pořádku, blok simuluje správně
- **-213**: nekompatibilita rozměrů matic stavového modelu
- **-510**: úloha je špatně podmíněná (některá z pracovních matic je singulární nebo blízka singulární matici)
- **xxx**: chybový kód **xxx** systému REXYGEN, více viz přílohu C

Tento blok propaguje kvalitu signálu. Více informací je uvedeno v sekci 1.4.

Vstup

R1	Reset bloku	Bool
HLD	Podržení aktuálního stavu modelu	Bool
u1..u16	Analogový vstupní signál	Double (F64)

Parametr

UD	Příznak použití matice D_c	Bool
is	Řád Padéovy aproximace	↓0 ↑4 ⊙2 Long (I32)
eps	Přesnost aproximace	↓0.0 ↑1.0 ⊙1e-15 Double (F64)
Ac	Matice A spojitého modelu	Double (F64)
	⊙[-0.36 -1.24 -0.18; 1 0 0; 0 1 0]	
Bc	Matice B spojitého modelu	⊙[0.5; 0; 0] Double (F64)
Cc	Matice C spojitého modelu	⊙[0.12 0.48 0.36] Double (F64)
Dc	Matice D spojitého modelu	⊙[0] Double (F64)
x0	Počáteční hodnota stavu x	⊙[0; 0; 0] Double (F64)

Výstup

iE

Kód chyby

Error

y1..y16

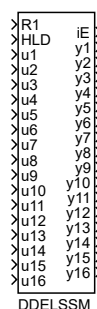
Analogový výstupní signál

Double (F64)

DDELSSM – Stavový model diskrétního lineárního systému s dopravním zpožděním

Symbol bloku

Licence: [ADVANCED](#)



Popis funkce

Funkční blok DDELSSM (Discrete State Space Model with time DELay) simuluje chování lineárního diskrétního systému s dopravním zpožděním *del* ve stavové reprezentaci:

$$\begin{aligned} x(k+1) &= A_d x(k) + B_d u(k-d), \quad x(0) = x_0 \\ y(k) &= C_d x(k) + D_d u(k), \end{aligned}$$

kde k je krok simulace, $x(k) \in \mathbb{R}^n$ je vektor stavu, $x_0 \in \mathbb{R}^n$ je počáteční hodnota vektoru stavu, $u(k) \in \mathbb{R}^m$ je vektor vstupu, $y(k) \in \mathbb{R}^p$ je vektor výstupu. Matice $A_d \in \mathbb{R}^{n \times n}$ určuje dynamiku systému, matice $B_d \in \mathbb{R}^{n \times m}$ určuje působení vstupu na stav systému, matice $C_d \in \mathbb{R}^{p \times n}$ určuje působení stavu na výstup systému a matice $D_d \in \mathbb{R}^{p \times m}$ určuje přímé působení vstupu na výstup systému. Pokud je `UD=off`, matice D_d se při simulaci nepoužívá (chová se jako by byla nulová). Počet kroků zpoždění d je největší celé číslo takové, že $d.T \leq del$, kde T je perioda spouštění bloku.

Všechny matice se zadávají stejným způsobem jako v systému Matlab, tj. celá matice je uzavřena v hranatých závorkách, zadává se po řádcích, jednotlivé prvky v řádku se oddělují mezerou, jednotlivé řádky středníkem. Pro oddělení desetinné části čísla se používá tečka. Vektor x_0 je sloupcový, proto se všechny jeho prvky oddělují středníkem (každý prvek je na samostatném řádku).

Při simulaci v reálném čase se v každém okamžiku spuštění bloku vždy vypočte jeden krok podle diskrétního stavového modelu uvedeného výše. Výstupy simulovaného systému $y_1 \dots y_{16}$ reprezentují stav systému $x(t)$ a pro danou simulaci se používá prvních p výstupů, kde p je počet řádků matice C_d .

Výstup `iE` je celočíselný a obsahuje informaci o průběhu simulace:

- **0:** vše v pořádku, blok simuluje správně

- **-213:** nekompatibilita rozměrů matic stavového modelu
- **xxx:** chybový kód xxx systému REXYGEN, více viz přílohu C

Tento blok propaguje kvalitu signálu. Více informací je uvedeno v sekci 1.4.

Vstup

R1	Reset bloku	Bool
HLD	Podržení aktuálního stavu modelu	Bool
u1..u16	Analogový vstupní signál	Double (F64)

Parametr

UD	Příznak použití matice Dd	Bool
del	Dopravní zpoždění modelu [s]	↓0.0 Double (F64)
Ad	Matice A diskrétního modelu	Double (F64)
	⊙[0.235700090 -0.904208075 -0.120785644; 0.671031354 0.477271377 -0.072129196; 0.000000000 0.000000000 0.000000000]	
Bd	Matice B diskrétního modelu	Double (F64)
	⊙[0.335515677; 0.200358878; 0.071773902]	
Cd	Matice C diskrétního modelu	⊙[0.12 0.48 0.36] Double (F64)
Dd	Matice D diskrétního modelu	⊙[0] Double (F64)
x0	Počáteční hodnota stavu x	⊙[0; 0; 0] Double (F64)

Výstup

iE	Kód chyby	Error
y1..y16	Analogový výstupní signál	Double (F64)

DFIR – Filtr s konečnou impulzní odezvou

Symbol bloku

Licence: [ADVANCED](#)



Popis funkce

Blok **DFIR** je filtr, jehož impulsní odezva (nebo odezva na jakýkoliv vstup konečné délky) je konečné délky, protože se ustálí na nule v konečném čase. Výpočet probíhá ve formě konvolučního integrálu (sumy) - impulsní charakteristika je zadána v poli **hk** již v diskretizované podobě pro správnou periodu.

Tento blok propaguje kvalitu signálu. Více informací je uvedeno v sekci [1.4](#).

Vstup

RST	Reset bloku	Bool
HLD	Pozastavení	Bool
u0	Počáteční hodnota po resetu modelu	Double (F64)

Parametr

nmax	Rezervovaná paměť pro pole	↓10 ↑100000000 ⊙100	Long (I32)
hk	Diskretizovaná impulzní odezva	⊙[0.6 0.3 0.1]	Double (F64)

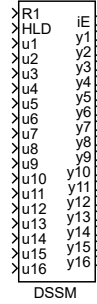
Výstup

y	Analogový výstupní signál	Double (F64)
RDY	Příznak platnosti/připravenosti výstupu	Bool

DSSM – Stavový model diskrétního lineárního systému

Symbol bloku

Licence: [ADVANCED](#)



Popis funkce

Funkční blok **DSSM** (Discrete State Space Model) simuluje chování lineárního diskrétního systému ve stavové reprezentaci:

$$\begin{aligned} x(k+1) &= A_d x(k) + B_d u(k), \quad x(0) = x_0 \\ y(k) &= C_d x(k) + D_d u(k), \end{aligned}$$

kde k je krok simulace, $x(k) \in \mathbb{R}^n$ je vektor stavu, $x_0 \in \mathbb{R}^n$ je počáteční hodnota vektoru stavu, $u(k) \in \mathbb{R}^m$ je vektor vstupu, $y(k) \in \mathbb{R}^p$ je vektor výstupu. Matice $A_d \in \mathbb{R}^{n \times n}$ určuje dynamiku systému, matice $B_d \in \mathbb{R}^{n \times m}$ určuje působení vstupu na stav systému, matice $C_d \in \mathbb{R}^{p \times n}$ určuje působení stavu na výstup systému a matice $D_d \in \mathbb{R}^{p \times m}$ určuje přímé působení vstupu na výstup systému. Pokud je **UD=off**, matice D_d se při simulaci nepoužívá (chová se jako by byla nulová).

Všechny matice se zadávají stejným způsobem jako v systému Matlab, tj. celá matice je uzavřena v hranatých závorkách, zadává se po řádcích, jednotlivé prvky v řádku se oddělují mezerou, jednotlivé řádky středníkem. Pro oddělení desetinné části čísla se používá tečka. Vektor x_0 je sloupcový, proto se všechny jeho prvky oddělují středníkem (každý prvek je na samostatném řádku).

Při simulaci v reálném čase se v každém okamžiku spuštění bloku vždy vypočte jeden krok podle diskrétního stavového modelu uvedeného výše. Výstupy simulovaného systému $y_1 \dots y_{16}$ reprezentují stav systému $x(t)$ a pro danou simulaci se používá prvních p výstupů, kde p je počet řádků matice C_d .

Výstup **iE** je celočíselný a obsahuje informaci o průběhu simulace:

- **0**: vše v pořádku, blok simuluje správně
- **-213**: nekompatibilita rozměrů matic stavového modelu
- **xxx**: chybový kód **xxx** systému REXYGEN, více viz přílohu [C](#)

Tento blok propaguje kvalitu signálu. Více informací je uvedeno v sekci [1.4](#).

Vstup

R1	Reset bloku	Bool
HLD	Podržení aktuálního stavu modelu	Bool
u1..u16	Analogový vstupní signál	Double (F64)
u	Analogový vstupní signál	Double (F64)

Parametr

UD	Příznak použití matice Dd	Bool
Ad	Matice A diskrétního modelu $\odot [0.235700090 \ -0.904208075 \ -0.120785644; \ 0.671031354 \ 0.477271377 \ -0.072129196; \ 0.40071771]$	Double (F64)
Bd	Matice B diskrétního modelu $\odot [0.335515677; \ 0.200358878; \ 0.071773902]$	Double (F64)
Cd	Matice C diskrétního modelu $\odot [0.12 \ 0.48 \ 0.36]$	Double (F64)
Dd	Matice D diskrétního modelu $\odot [0]$	Double (F64)
x0	Počáteční hodnota stavu x $\odot [0; \ 0; \ 0]$	Double (F64)

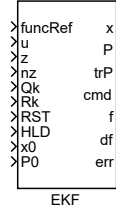
Výstup

iE	Kód chyby	Error
y1..y16	Analogový výstupní signál	Double (F64)

EKF – Rozšířený (nelineární) Kalmanův filtr

Symbol bloku

Licence: [MODEL](#)



Popis funkce

Funkční blok implementuje algoritmus nelineárního rekonstruktoru stavu známého jako Rozšířený Kalmanův filtr. Cílem je poskytnout odhad přímo neměřitelných stavových veličin nelineárního dynamického systému popsaného stavovou a výstupní rovnicí ve tvaru

$$\begin{aligned}\frac{dx(t)}{dt} &= f(x(t), u(t)) + w(t), \\ y(t) &= h(x(t), u(t)) + v(t)\end{aligned}$$

pro případ spojitého času a

$$\begin{aligned}x(k+1) &= f(x(k), u(k)) + w(k), \\ y(k) &= h(x(k), u(k)) + v(k)\end{aligned}$$

pro případ diskrétního systému. Veličina x vyjadřuje stavový vektor, u vektor vstupu, y vektor výstupu. Veličiny w a v označují náhodný šum modelu a pozorování. Předpokládá se, že se jedná o náhodné procesy s nulovou střední hodnotou a Gaussovým rozdělením hustoty pravděpodobnosti, definovaným kovariančními maticemi Q_k a R_k , které jsou zadávány jako parametry bloku:

$$\begin{aligned}w(t) &\sim \mathcal{N}(0, Q_k), \\ v(t) &\sim \mathcal{N}(0, R_k).\end{aligned}$$

Rozšířený Kalmanův filtr je nelineární verze algoritmu Kalmanova filtru pracující na principu linearizace stavové a výstupní rovnice v okolí aktuálního pracovního bodu. Jde o algoritmus typu prediktor-korektor, který střídá fázi predikce stavu v otevřené smyčce s využitím modelu a korekce (filtrace) na základě přímo měřených pozorování. Vektor výstupů-pozorování může být dodáván asynchronně vůči periodickému běhu algoritmu filtrace v libovolných okamžicích spouštění bloku.

Krok predikce je vykonáván každou periodu běhu bloku a řeší stavovou rovnici technikami numerické integrace, počínaje zadaným stavem $x0$ a počáteční kovariancí $P0$.

Volbou parametru *solver* uživatel vybírá numerickou metodu integrace příslušné vektorové diferenciální rovnice. Pro speciální případ volby *solver* = 1 algoritmus přechází na diskrétní variantu modelu a numerická integrace se redukuje na pouhé vyhodnocení pravé strany rekurze definované stavovou diferenční rovnicí $x(k+1) = f(x(k), u(k))$. Kromě vektoru stavu je v čase propagována také příslušná kovarianční matice P , která reprezentuje rozptyl odhadu stavu a jeho korelace mezi složkami. Odhad stavu je modelován jako náhodná veličina s normálním rozdělením:

$$\hat{x}(t) \sim \mathcal{N}(x, P).$$

Více detailů o jednotlivých numerických metodách lze nalézt v dokumentaci k bloku [NSSM](#).

Krok filtrace je proveden vždy když je v daném okamžiku spuštění bloku na vstup přivedena hodnota $nz > 0$. Toto signalizuje dostupný vektor měření na vstupu z , který je následně použit pro opravu odhadu stavu a jeho kovariance. Je možné kombinovat více pravých stran výstupní rovnice pomocí kooperujícího bloku [REXLANG](#). Toto může být užitečné v aplikacích s větším počtem senzorů, které dodávají data s různou periodou vzorkování nebo nepravidelně oproti periodickému spouštění bloku. Pro nastavení $nz = 0$ signalizuje uživatelský algoritmus bloku nedostupnost měření a v dané periodě je provedena extrapolace stavu pozorovaného systému na základě modelu.

Kalmanův filtr obecně není optimální rekonstruktor stavu ve smyslu minimalizace střední kvadratické chyby odhadu. Nicméně, v praktických úlohách s dostatečně hladkou nelineární dynamikou systému poskytuje odhady stavu v rozumné kvalitě a je považován za de facto standard v oblasti nelineární filtrace. V případě zadání lineární pravé strany stavové a výstupní rovnice přechází algoritmus odhadu na standardní Kalmanův filtr, který je již optimální pro danou stochastickou formulaci problému odhadu stavu lineárního systému.

Tento blok nepropaguje kvalitu signálu. Více informací je uvedeno v sekci [1.4](#).

Vstup

funcRef	Odkaz na spolupracující blok REXLANG	Reference
u	Vektor vstupů modelu	Reference
z	Vektor výstupů (měření) modelu	Reference
nz	Číslo sady měřených výstupů	↓1 Long (I32)
Qk	Kovarianční matice stavového šumu	Reference
Rk	Kovarianční matice výstupního šumu	Reference
RST	Reset bloku	Bool
HLD	Pozastavení	Bool
x0	Vektor počáteční hodnoty stavu modelu	Reference
P0	Počáteční hodnota kovarianční matice modelu	Reference

Parametr

nmax	Rezervovaná paměť pro výstupní matici (celkový počet prvků) ↓5 ↑10000 ⊙20	Long (I32)
solver	Metoda numerické integrace 1 Diskrétní model 2 eulerova metoda (1. řád) 3 metoda Adams-Bashforth 2. řádu 4 metoda Adams-Bashforth 3. řádu 5 metoda Adams-Bashforth 4. řádu 6 metoda Adams-Bashforth 5. řádu 7 Runge-Kutha 4. řádu 8 implicitní Euler 9 implicitní Euler (více iterací) 10 implicitní Adams-Multon 2. řádu 11 implicitní Adams-Multon 2. řádu (více iterací) 12 implicitní Adams-Multon 3. řádu 13 implicitní Adams-Multon 3. řádu (více iterací) 14 implicitní RadauIIA 2. řádu 15 implicitní RadauIIA 2. řádu (více iterací) 16 — 17 — 18 — 19 —	⊙2 Long (I32)

Výstup

x	Vektor stavu modelu	Reference
P	Kovarianční matice stavu modelu	Reference
trP	Stopa kovarianční matice	Reference
cmd	Číslo příkazu navázaného bloku REXLANG	Long (I32)
f	Odkaz na vektor nastavovaný navázaným blokem REXLANG	Reference
df	Odkaz na matici nastavovanou navázaným blokem REXLANG	Reference
err	Kód chyby (0 bez chyby, jinak popis v system logu)	Long (I32)

FOPDT – Model systému 1. řádu s dopravním zpožděním

Symbol bloku

Licence: [STANDARD](#)



Popis funkce

Blok FOPDT realizuje diskretní simulátor lineárního systému prvního řádu s přídavným dopravním zpožděním, který je popsán následující přenosovou funkcí:

$$P(s) = \frac{k_0}{(\tau \cdot s + 1)} \cdot e^{-\text{del} \cdot s}$$

Diskretní simulace používá přesnou diskretizaci přenosu $P(s)$ pro periodu T_S , s níž je blok FOPDT spouštěn.

Tento blok propaguje kvalitu signálu. Více informací je uvedeno v sekci [1.4](#).

Vstup

u	Analogový vstupní signál	Double (F64)

Parametr

k0	Statické zesílení	⊙1.0	Double (F64)
del	Dopravní zpoždění [s]		Double (F64)
tau	Časová konstanta	⊙1.0	Double (F64)
nmax	Rezervovaná paměť pro pole	↓10 ↑10000000 ⊙1000	Long (I32)

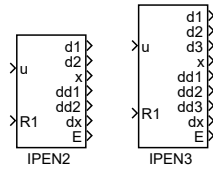
Výstup

y	Analogový výstupní signál	Double (F64)

IPEN2, IPEN3 – N-ramenné inverzní kyvadlo na vozíku - Fyzikální parametry

Symbole bloků

Licence: [MODEL](#)



Popis funkce

Bloky IPEN2 a IPEN3 simulují dynamiku dvojitého a trojitého inverzního kyvadla na vozíku. Tyto modely umožňují uživatelům provádět experimenty s různými řídicími strategiemi, čímž jsou vhodné jak pro vzdělávací účely, tak pro vědecký výzkum.

Hlavním vstupem do modelů je analogový signál u , jehož interpretace závisí na nastavení parametru IACC:

- pro IACC=on je vstup u považován za sílu působící na vozík [N],
- pro IACC=off modely předpokládají, že vstup představuje rychlost [m/s].

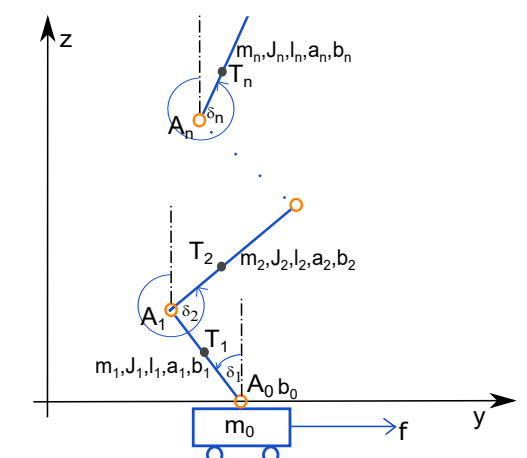
Signál R1 slouží k resetování každého modelu do jeho počáteční konfigurace.

Oba modely lze detailně konfigurovat pomocí série parametrů, které odpovídají fyzikálním charakteristikám systému. Tyto parametry zahrnují relativní polohu těžiště kyvadel a , moment setrvačnosti J , délku l a hmotnost kyvadel m , koeficienty tlumení b a počáteční stav systému (polohy d_0 , rychlosti dd_0). Schematické znázornění systému s parametry je na obrázku níže. Většina parametrů má jasnou definici, relativní poloha těžiště i -tého kyvadla a_i je dána vztahem

$$|A_{i-1}T_i| = a_i l_i,$$

kde A_{i-1} je pozice předchozího kloubu, T_i je pozice těžiště kyvadla a l_i je délka kyvadla.

Výpočet modelů je v souladu s matematickým modelem a fyzikálními parametry, které jsou podrobně popsány v literatuře [9]. Bloky [IPEN2pu](#) a [IPEN3pu](#) jsou využívány pro simulaci modelů inverzních kyvadel s dynamickými parametry.



Tento blok nepropaguje kvalitu signálu. Více informací je uvedeno v sekci 1.4.

Vstup

u	Analogový vstupní signál	Double (F64)
R1	Reset bloku	Bool

Parametr

a1	Relativní poloha těžiště	⊙1.0	Double (F64)
a2	Relativní poloha těžiště	⊙1.0	Double (F64)
a3	Relativní poloha těžiště	⊙1.0	Double (F64)
J1	Moment setrvačnosti kyvadla [kg.m ²]	⊙1.0	Double (F64)
J2	Moment setrvačnosti kyvadla [kg.m ²]	⊙1.0	Double (F64)
J3	Moment setrvačnosti kyvadla [kg.m ²]	⊙1.0	Double (F64)
l1	Délka kyvadla [m]	⊙1.0	Double (F64)
l2	Délka kyvadla [m]	⊙1.0	Double (F64)
l3	Délka kyvadla [m]	⊙1.0	Double (F64)
m1	Hmotnost kyvadla [kg]	⊙1.0	Double (F64)
m2	Hmotnost kyvadla [kg]	⊙1.0	Double (F64)
m3	Hmotnost kyvadla [kg]	⊙1.0	Double (F64)
m0	Hmotnost vozíku [kg]	⊙1.0	Double (F64)
b1	Koeficient tlumení kyvadla	⊙1.0	Double (F64)
b2	Koeficient tlumení kyvadla	⊙1.0	Double (F64)
b3	Koeficient tlumení kyvadla	⊙1.0	Double (F64)
b0	Koeficient tlumení vozíku	⊙1.0	Double (F64)
d1_0	Počáteční poloha kyvadla [rad]	⊙1.0	Double (F64)
d2_0	Počáteční poloha kyvadla [rad]	⊙1.0	Double (F64)
d3_0	Počáteční poloha kyvadla [rad]	⊙1.0	Double (F64)
x_0	Počáteční poloha vozíku [m]	⊙1.0	Double (F64)

dd1_0	Počáteční úhlová rychlost kyvadla [rad/s]	⊙1.0	Double (F64)
dd2_0	Počáteční úhlová rychlost kyvadla [rad/s]	⊙1.0	Double (F64)
dd3_0	Počáteční úhlová rychlost kyvadla [rad/s]	⊙1.0	Double (F64)
dx_0	Počáteční rychlost vozíku [m/s]	⊙1.0	Double (F64)
IACC	on=Vstup u je síla, off=Vstup u je rychlost		Bool

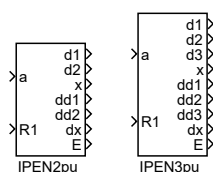
Výstup

d1	Poloha kyvadla [rad]	Double (F64)
d2	Poloha kyvadla [rad]	Double (F64)
d3	Poloha kyvadla [rad]	Double (F64)
x	Poloha vozíku [m]	Double (F64)
dd1	Úhlová rychlost kyvadla [rad/s]	Double (F64)
dd2	Úhlová rychlost kyvadla [rad/s]	Double (F64)
dd3	Úhlová rychlost kyvadla [rad/s]	Double (F64)
dx	Rychlost vozíku [m/s]	Double (F64)
E	Příznak chyby	Bool

IPEN2pu, IPEN3pu – N-ramenné inverzní kyvadlo na vozíku - Dynamické parametry

Symbole bloků

Licence: [MODEL](#)



Popis funkce

Bloky IPEN2pu a IPEN3pu simulují dynamiku dvojitého a trojitého inverzního kyvadla na vozíku. Tyto modely umožňují uživatelům provádět experimenty s různými řídicími strategiemi, čímž jsou vhodné jak pro vzdělávací účely, tak pro vědecký výzkum.

Hlavním vstupem modelů je analogový signál **a** reprezentující zrychlení vozíku $[m/s^2]$. Pro resetování modelů do počátečního stavu slouží signál **R1**.

Oba modely lze detailně konfigurovat pomocí dynamických parametrů systému **p1** - **p11** a počátečního stavu systému (polohy **d_0**, rychlosti **dd_0**). Dynamické parametry lze určit pomocí fyzikálních parametrů definovaných v blocích [IPEN2](#), [IPEN3](#) a rovnic uvedených níže. Detaily modelů jsou podrobně popsány v literatuře [9].

Pro model IPEN2pu jsou dynamické parametry definovány následovně:

$$p_1 = \frac{(m_1 a_1^2 + m_2) l_1^2 + J_1}{m_2 a_2 l_2 l_1}, \quad p_2 = \frac{a_2^2 l_2^2 m_2 + J_2}{m_2 a_2 l_2 l_1}, \quad p_3 = \frac{m_1 a_1 + m_2}{m_2 a_2 l_2},$$

$$p_4 = \frac{1}{l_1}, \quad p_5 = \frac{b_1}{m_2 a_2 l_2 l_1}, \quad p_6 = \frac{b_2}{m_2 a_2 l_2 l_1}.$$

Pro model IPEN3pu jsou dynamické parametry definovány jako:

$$p_1 = \frac{l_1(m_2 a_2 + m_3)}{a_3 l_3 m_3}, \quad p_2 = \frac{l_1}{l_2}, \quad p_3 = \frac{(m_1 a_1^2 + m_2 + m_3) l_1^2 + J_1}{a_3 l_3 m_3 l_2},$$

$$p_4 = \frac{(m_2 a_2^2 + m_3) l_2^2 + J_2}{a_3 l_3 m_3 l_2}, \quad p_5 = \frac{a_3^2 l_3^2 m_3 + J_3}{a_3 l_3 m_3 l_2}, \quad p_6 = \frac{l_1(m_1 a_1 + m_2 + m_3)}{a_3 l_3 m_3 l_2},$$

$$p_7 = \frac{a_2 m_2 + m_3}{a_3 l_3 m_3}, \quad p_8 = \frac{1}{l_2}, \quad p_9 = \frac{b_1}{a_3 l_3 m_3 l_2}, \quad p_{10} = \frac{b_2}{a_3 l_3 m_3 l_2}, \quad p_{11} = \frac{b_3}{a_3 l_3 m_3 l_2}.$$

Tento blok nepropaguje kvalitu signálu. Více informací je uvedeno v sekci [1.4](#).

Vstup

a	Zrychlení vozíku [m/s ²]	Double (F64)
R1	Reset bloku	Bool

Parametr

p1..p11	Dynamický parametr modelu (viz rovnice modelu)	⊙1.0 Double (F64)
d1_0	Počáteční poloha kyvadla [rad]	⊙1.0 Double (F64)
d2_0	Počáteční poloha kyvadla [rad]	⊙1.0 Double (F64)
d3_0	Počáteční poloha kyvadla [rad]	⊙1.0 Double (F64)
x_0	Počáteční poloha vozíku [m]	⊙1.0 Double (F64)
dd1_0	Počáteční úhlová rychlost kyvadla [rad/s]	⊙1.0 Double (F64)
dd2_0	Počáteční úhlová rychlost kyvadla [rad/s]	⊙1.0 Double (F64)
dd3_0	Počáteční úhlová rychlost kyvadla [rad/s]	⊙1.0 Double (F64)
dx_0	Počáteční rychlost vozíku [m/s]	⊙1.0 Double (F64)

Výstup

d1	Poloha kyvadla [rad]	Double (F64)
d2	Poloha kyvadla [rad]	Double (F64)
d3	Poloha kyvadla [rad]	Double (F64)
x	Poloha vozíku [m]	Double (F64)
dd1	Úhlová rychlost kyvadla [rad/s]	Double (F64)
dd2	Úhlová rychlost kyvadla [rad/s]	Double (F64)
dd3	Úhlová rychlost kyvadla [rad/s]	Double (F64)
dx	Rychlost vozíku [m/s]	Double (F64)
E	Příznak chyby	Bool

MDL – Model procesu

Symbol bloku

Licence: [STANDARD](#)



Popis funkce

Blok MDL realizuje diskretní simulátor spojitého systému s přenosem

$$F(s) = \frac{K_0 e^{-Ds}}{(\tau_1 s + 1)(\tau_2 s + 1)},$$

kde $K_0 > 0$ je statické zesílení **k0**, $D \geq 0$ je dopravní zpoždění **del** a $\tau_1, \tau_2 > 0$ jsou časové konstanty systému **tau1** a **tau2**.

Tento blok propaguje kvalitu signálu. Více informací je uvedeno v sekci [1.4](#).

Vstup

u	Analogový vstupní signál	Double (F64)

Parametr

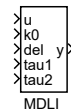
k0	Statické zesílení	⊙1.0	Double (F64)
del	Dopravní zpoždění [s]		Double (F64)
tau1	První časová konstanta	⊙1.0	Double (F64)
tau2	Druhá časová konstanta	⊙2.0	Double (F64)
nmax	Rezervovaná paměť pro pole	↓10 ↑100000000 ⊙1000	Long (I32)

Výstup

y	Analogový výstupní signál	Double (F64)

MDLI – Model procesu s proměnnými parametry

Symbol bloku

Licence: [STANDARD](#)

Popis funkce

Blok MDLI realizuje diskrétní simulátor spojitého systému s přenosem

$$F(s) = \frac{K_0 e^{-Ds}}{(\tau_1 s + 1)(\tau_2 s + 1)},$$

kde $K_0 > 0$ je statické zesílení **k0**, $D \geq 0$ je dopravní zpoždění **del** a $\tau_1, \tau_2 > 0$ jsou časové konstanty systému **tau1** a **tau2**. Na rozdíl od bloku [MDL](#) mohou být všechny parametry systému průběžně měněny ze vstupů bloku.

Tento blok propaguje kvalitu signálu. Více informací je uvedeno v sekci [1.4](#).

Vstup

u	Analogový vstupní signál	Double (F64)
k0	Statické zesílení	Double (F64)
del	Dopravní zpoždění [s]	Double (F64)
tau1	První časová konstanta	Double (F64)
tau2	Druhá časová konstanta	Double (F64)

Parametr

nmax	Rezervovaná paměť pro pole	↓10 ↑100000000 ⊕1000	Long (I32)

Výstup

y	Analogový výstupní signál	Double (F64)

MVD – Motorizovaný pohon ventilu

Symbol bloku

Licence: [STANDARD](#)



Popis funkce

Blok MVD je určen pro simulaci servoventilu. Vstup UP (DN) představuje binární povel pro otevírání (zavírání) ventilu konstantní rychlostí $1/t_v$, kde t_v je parametr bloku. Při UP = on (DN = on) otevírání probíhá až do úplného otevření $y = h_{lim}$ (úplného zavření $y = l_{lim}$) ventilu. Krajiní poloha otevření (zavření) je signalizována koncovým spínačem HS (LS). Počáteční poloha ventilu po spuštění je $y = y_0$. Jestliže UP = DN = on nebo UP = DN = off, pak se poloha ventilu nemění (ani nezavírá ani neotevřívá).

Tento blok propaguje kvalitu signálu. Více informací je uvedeno v sekci [1.4](#).

Vstup

UP	Otevřít	Bool
DN	Zavřít	Bool

Parametr

y_0	Počáteční poloha ventilu		Double (F64)
t_v	Čas přejezdu mezi polohami $y=0$ a $y=1$ [s]	⊖10.0	Double (F64)
h_{lim}	Horní mezní poloha (otevřeno)	⊖1.0	Double (F64)
l_{lim}	Dolní mezní poloha (zavřeno)		Double (F64)

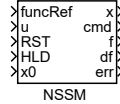
Výstup

y	Poloha ventilu	Double (F64)
HS	Horní koncový spínač	Bool
LS	Dolní koncový spínač	Bool

NSSM – Nelineární stavový model

Symbol bloku

Licence: [MODEL](#)



Popis funkce

Blok poskytuje řešení stavové a výstupní rovnice nelineárního dynamického systému popsaného soustavou diferenciálních rovnic

$$\begin{aligned}\frac{dx(t)}{dt} &= f(x(t), u(t)), \\ y(t) &= h(x(t), u(t))\end{aligned}$$

pro spojitý čas nebo

$$\begin{aligned}x(k+1) &= f(x(k), u(k)), \\ y(k) &= h(x(k), u(k)).\end{aligned}$$

pro diskrétní systém. Spojitá rovnice je interně diskretizována do tvaru

$$x(t) = F(x(t-T), u(t)),$$

kde T je perioda vzorkování NSSM bloku.

Metoda diskretizace a numerického řešení stavové rovnice závisí na volbě uživatelského parametru `solver`. K dispozici jsou metody jednokrokové (Runge-Kutta, Euler), vícekové (Adams-Bashforth) a implicitní (Adams-Moulton). U některých metod lze volit řád pro nalezení vhodného kompromisu mezi přesností a rychlostí výpočtu modelu.

Blok nepodporuje algoritmy řešení s variabilním krokem. Velikost časového kroku je vždy shodná s periodou spouštění funkčního bloku.

Nelineární vektorové funkce $f(x, u)$, $h(x, u)$ musí být implementovány v [REXLANG](#) bloku, který je připojen k bloku NSSM specifickým způsobem. Vstup `funcRef` bloku NSSM musí být připojen na výstup `y0` spolupracujícího [REXLANG](#) bloku a výstup `y0` nesmí být použit interně v kódu bloku [REXLANG](#). Výstupy `x`, `f` a `df` bloku NSSM musí být připojeny na vstupy bloku [REXLANG](#). Tyto vstupy musí být zpracovány v bloku [REXLANG](#) jako vstupní pole. Hlavní main funkce bloku [REXLANG](#) musí nastavovat hodnotu vektorové funkce $f(x, u)$ do vektoru `f` (tj. do vstupního pole, kam je `f` připojeno), matice $df(x, u)/dx$ se ukládá do matice `af`.

Blok NSSM volá funkci `main` bloku [REXLANG](#) pokud je potřeba vyhodnotit pravou stranu rovnice pro numerickou integraci (například Runge-Kutta metoda provádí 4 volání v každé periodě spuštění bloku s různými hodnotami argumentu `x`). Blok [REXLANG](#)

musí být ve schématu zakázán, aby se zabránilo jeho automatické spuštění systémem REXYGEN. Pokud je potřeba spouštění bloku [REXLANG](#) (například pro výpočet nelineární výstupní rovnice $y = h(x, u)$), je doporučeno zapojit výstup `cmd` bloku `NSSM` na vstup bloku [REXLANG](#) pro rozlišení mezi voláním z `NSSM` bloku pro numerickou integraci stavové rovnice (`cmd = 0`) a voláním ze systému REXYGEN system (`cmd = -1`).

Poznámky:

- výpočet Jakobiánu $df(x, u)/dx$ je nutný jen pro implicitní metody.
- velikost pole `x` (a tím také `f`, `df`) je definováno dimenzí pole `x0`. Změna velikosti je dovolena pouze při aktivaci vstupu `reset` (`RST`).
- `solver=1: discrete` signalizuje diskrétní stavovou rovnici s funkcí `f` označující pravou stranu příslušné diferenční rovnice. Tento režim nevyžaduje numerickou integraci a algoritmus se redukuje na prosté vyhodnocení pravé strany definované rekurzí; tento režim v principu nevyžaduje použití `NSSM` bloku (lze řešit přímo pomocí [REXLANG](#)); tento režim je zachován zejména z důvodu kompatibility s příbuzným blokem [EKF](#).
- pro blok `NSSM` je nutné připojení výstupu `cmd`, protože `cmd>0` indikuje počet měření, které musí vracet blok [REXLANG](#) při vyhodnocení $f = h(x, u)$, $df = dh(x, u)/dx$.

Tento blok nepropaguje kvalitu signálu. Více informací je uvedeno v sekci [1.4](#).

Vstup

<code>funcRef</code>	Odkaz na spolupracující blok REXLANG	Reference
<code>u</code>	Vektor vstupů modelu	Reference
<code>RST</code>	Reset bloku	Bool
<code>HLD</code>	Pozastavení	Bool
<code>x0</code>	Vektor počáteční hodnoty stavu modelu	Reference

Parametr

<code>nmax</code>	Rezervovaná paměť pro výstupní matici (celkový počet prvků) ↓5 ↑10000 ⊙20	Long (I32)

<code>solver</code>	Metoda numerické integrace	⊙2 Long (I32)
1	 Diskrétní model	
2	 eulerova metoda (1. řád)	
3	 metoda Adams-Bashforth 2. řádu	
4	 metoda Adams-Bashforth 3. řádu	
5	 metoda Adams-Bashforth 4. řádu	
6	 metoda Adams-Bashforth 5. řádu	
7	 Runge-Kutha 4. řádu	
8	 implicitní Euler	
9	 implicitní Euler (více iterací)	
10	 implicitní Adams-Multon 2. řádu	
11	 implicitní Adams-Multon 2. řádu (více iterací)	
12	 implicitní Adams-Multon 3. řádu	
13	 implicitní Adams-Multon 3. řádu (více iterací)	
14	 implicitní RadauIIA 2. řádu	
15	 implicitní RadauIIA 2. řádu (více iterací)	
16	 —	
17	 —	
18	 —	
19	 —	

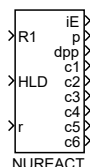
Výstup

<code>x</code>	Vektor stavu modelu	Reference
<code>y</code>	Výstupní vektor modelu	Reference
<code>cmd</code>	Číslo příkazu navázaného bloku REXLANG	Long (I32)
<code>f</code>	Odkaz na vektor nastavovaný navázaným blokem REXLANG	Reference
<code>df</code>	Odkaz na matici nastavovanou navázaným blokem REXLANG	Reference
<code>err</code>	Kód chyby (0 bez chyby, jinak popis v system logu)	Long (I32)

NUREACT – Model of nuclear reactor

Symbol bloku

Licence: [MODEL](#)



Popis funkce

Popis tohoto bloku ještě není k dispozici. Níže naleznete částečný popis vstupů, výstupů a parametrů bloku. Kompletní popis bloku bude k dispozici v dalších revizích dokumentace.

Tento blok nepropaguje kvalitu signálu. Více informací je uvedeno v sekci [1.4](#).

Vstup

R1	Reset bloku	Bool
HLD	Podržení aktuálního stavu modelu	Bool
r	Reaktivita []	Double (F64)

Parametr

p0	Počáteční výkon reaktoru [W]	⊙1.0	Double (F64)
b1..b6	Parametr modelu beta(viz rovnice modelu)	⊙1.0	Double (F64)
l1..l6	Parametr modelu lambda (viz rovnice modelu)	⊙1.0	Double (F64)
tau	Časová konstanta [s]	⊙1.0	Double (F64)

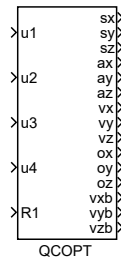
Výstup

iE	Kód chyby	Bool
p	Tepelný výkon reaktoru [W]	Double (F64)
dpp	Derivace tepelného výkonu reaktoru [W]	Double (F64)
c1..c6	Koncentrace neutronů (viz rovnice modelu)	Double (F64)

QCOPT – Model dronu (kvadrukoptéry)

Symbol bloku

Licence: [MODEL](#)



Popis funkce

Popis tohoto bloku ještě není k dispozici. Níže naleznete částečný popis vstupů, výstupů a parametrů bloku. Kompletní popis bloku bude k dispozici v dalších revizích dokumentace.

Tento blok nepropaguje kvalitu signálu. Více informací je uvedeno v sekci [1.4](#).

Vstup

u1..u4	požadovaná hodnota motoru	Double (F64)
R1	Reset bloku	Bool

Parametr

m	Celková hmotnost [kg]	⊙1.0	Double (F64)
g	Gravitační zrychlení [m/s ²]	⊙1.0	Double (F64)
KT	Koeficient tahu motoru [N/u]	⊙1.0	Double (F64)
KTT	Koeficient točivého momentu motoru [Nm/u]	⊙1.0	Double (F64)
Ixx	Prvek matice setrvačnosti	⊙1.0	Double (F64)
Iyy	Prvek matice setrvačnosti	⊙1.0	Double (F64)
Izz	Prvek matice setrvačnosti	⊙1.0	Double (F64)
pM	Matice poloh rotorů [x1 y1 z1; x2 y2 z2; x3 y3 z3; x4 y4 z4]	⊙1.0	Double (F64)

Výstup

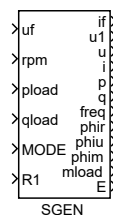
sx	Poloha těžiště [m]	Double (F64)
sy	Poloha těžiště [m]	Double (F64)
sz	Poloha těžiště [m]	Double (F64)
vx	Rychlost těžiště [m/s]	Double (F64)
vy	Rychlost těžiště [m/s]	Double (F64)

vz	Rychlost těžiště [m/s]	Double (F64)
ax	Natočení (eulerovy úhly) [rad]	Double (F64)
ay	Natočení (eulerovy úhly) [rad]	Double (F64)
az	Natočení (eulerovy úhly) [rad]	Double (F64)
ox	Rychlost otáčení (derivace eulerovo úhlů) [rad/s]	Double (F64)
oy	Rychlost otáčení (derivace eulerovo úhlů) [rad/s]	Double (F64)
oz	Rychlost otáčení (derivace eulerovo úhlů) [rad/s]	Double (F64)
vxb	Velocity in the body frame	Double (F64)
vyb	Velocity in the body frame	Double (F64)
vzb	Velocity in the body frame	Double (F64)

SGEN – Model synchronního generátoru

Symbol bloku

Licence: [MODEL](#)



Popis funkce

Popis tohoto bloku ještě není k dispozici. Níže naleznete částečný popis vstupů, výstupů a parametrů bloku. Kompletní popis bloku bude k dispozici v dalších revizích dokumentace.

Tento blok nepropaguje kvalitu signálu. Více informací je uvedeno v sekci [1.4](#).

Vstup

uf	Budící napětí[V]	Double (F64)
rpm	Otáčky za minutu	Double (F64)
pload	Odebíraný činný výkon [W]	Double (F64)
qload	Odebíraný jalový příkon [VAr]	Double (F64)
MODE	1=generator přifázovaný k síti, 0=ostrovní režim	Bool
R1	Reset modelu	Bool

Parametr

pn	Jmenovitý výkon [VA]	⊙1.0	Double (F64)
un	Jmenovité efektivní napětí [V]	⊙1.0	Double (F64)
fn	Jmenovitá frekvence [Hz]	⊙1.0	Double (F64)
ifn	Jmenovitý budící proud [A]		Double (F64)
pp	Počet polpárů	↓1 ↑10000 ⊙2	Long (I32)
ZM	1=režim konstantní impedance, 0=režim konstantního výkonu		Bool
ra	Odpor statorového vinutí[p.u.]	⊙0.011	Double (F64)
rf	Odpor budícího vinutí[p.u.]	⊙0.0006	Double (F64)
r1d	Odpor tlumícího vinutí v ose D [p.u.]	⊙0.0354	Double (F64)
r1q	Odpor tlumícího vinutí v ose Q [p.u.]	⊙0.0428	Double (F64)
ld	Indukčnost statorového vinutí v ose D [p.u.]	⊙1.05	Double (F64)
lq	Indukčnost statorového vinutí v ose Q [p.u.]	⊙0.7	Double (F64)
lad	Indukčnost tlumícího vinutí v ose D [p.u.]	⊙0.9	Double (F64)
laq	Indukčnost tlumícího vinutí v ose Q [p.u.]	⊙0.55	Double (F64)

lff	Indukčnost budícího vinutí [p.u.]	⊙1.571	Double (F64)
lf1d	Vzájemná indukčnost budícího a tlumícího vinutí [p.u.]	⊙0.9	Double (F64)
l11d	Vzájemná indukčnost statorového a tlumícího vinutí v ose D [p.u.]	⊙1.1	Double (F64)
l11q	Vzájemná indukčnost statorového a tlumícího vinutí v ose Q [p.u.]	⊙0.8067	Double (F64)

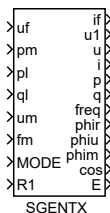
Výstup

if	Budící proud [A]	Double (F64)
u1	Okamžité napětí 1.fáze [V]	Double (F64)
u	Efektivní (fázové) napětí [V]	Double (F64)
i	Efektivní (fázové) proud [A]	Double (F64)
p	Odebíraný činný elektrický výkon [W]	Double (F64)
q	Odebíraný jalový příkon [VAr]	Double (F64)
freq	Frekvence[Hz]	Double (F64)
phir	Úhel natočení rotoru [rad]	Double (F64)
phiu	Zátěžový úhel [rad]	Double (F64)
phim	Úhel pro fázování [rad]	Double (F64)
mload	Kroutící moment hřídele [Nm]	Double (F64)
E	Příznak chyby	Bool

SGENTX – Model synchronního generátoru

Symbol bloku

Licence: [MODEL](#)



Popis funkce

Popis tohoto bloku ještě není k dispozici. Níže naleznete částečný popis vstupů, výstupů a parametrů bloku. Kompletní popis bloku bude k dispozici v dalších revizích dokumentace.

Tento blok nepropaguje kvalitu signálu. Více informací je uvedeno v sekci [1.4](#).

Vstup

uf	Budící napětí [p.u.]	Double (F64)
pm	Kroutící moment turbíny/motoru [p.u.]	Double (F64)
pload	Odebíraný činný výkon [p.u.]	Double (F64)
qload	Odebíraný jalový příkon [p.u.]	Double (F64)
MODE	1=generator přifázovaný k síti, 0=ostrovní režim	Bool
R1	Reset modelu	Bool

Parametr

um	Napětí sítě [p.u.]	⊙0.011	Double (F64)
fm	Kmitočet sítě [p.u.]	⊙0.011	Double (F64)
pn	Jmenovitý výkon [VA]	⊙10000000.0	Double (F64)
un	Jmenovité efektivní napětí [V]	⊙6000.0	Double (F64)
fn	Jmenovitá frekvence [Hz]	⊙1.0	Double (F64)
pp	Počet pólů	↓1 ↑1000 ⊙2	Long (I32)
ZM	1=režim konstantní impedance, 0=režim konstantního výkonu		Bool
J	Moment setrvačnosti [p.u.]	⊙0.0006	Double (F64)
Tdoi	1. časová konstanta (transientní) v ose D (na prázdko) [p.u.]	⊙0.0354	Double (F64)
Tdoi i	2. časová konstanta (subtransientní) v ose D (na prázdko) [p.u.]	⊙0.0354	Double (F64)
Tqoi	1. časová konstanta (transientní) v ose Q (na prázdko) [p.u.]	⊙0.0354	Double (F64)

Tqoii	2. časová konstanta (subtransientní) v ose Q (na prázdkno) [p.u.]	Double (F64)
	⊙0.0354	
Xd	Reaktance v ose D [p.u.]	Double (F64)
Xdi	Transientní reaktance v ose D [p.u.]	Double (F64)
Xdii	Subtransientní reaktance v ose D [p.u.]	Double (F64)
Xq	Reaktance v ose Q [p.u.]	Double (F64)
Xqi	Transientní reaktance v ose Q [p.u.]	Double (F64)
Xqii	Subtransientní reaktance v ose Q [p.u.]	Double (F64)
r	Odpor statorového vinutí [p.u.]	Double (F64)
Xmd	Křížová reaktance buzení [p.u.]	Double (F64)
rf	Odpor budícího vinutí [p.u.]	Double (F64)
rm	Odpor vedení do sítě [p.u.]	Double (F64)
xm	Reaktance vedení do sítě [p.u.]	Double (F64)

Výstup

if	Budící proud [p.u.]	Double (F64)
u1	Okamžité napětí 1.fáze [p.u.]	Double (F64)
u	Efektivní (fázové) napětí [p.u.]	Double (F64)
i	Efektivní (fázový) proud [p.u.]	Double (F64)
p	Generovaný činný elektrický výkon [p.u.]	Double (F64)
q	Generovaný reaktivní příkon [p.u.]	Double (F64)
freq	Frekvence [p.u.]	Double (F64)
phir	Úhel natočení rotoru [rad]	Double (F64)
phiu	Zátěžový úhel [rad]	Double (F64)
phim	Úhel pro fázování [rad]	Double (F64)
rpm	Otáčky za minutu	Double (F64)
cos	Účinník	Double (F64)
E	Příznak chyby	Bool

SOPDT – Model systému 2. řádu s dopravním zpožděním

Symbol bloku

Licence: [STANDARD](#)



Popis funkce

Blok SOPDT realizuje diskretní simulátor lineárního systému druhého řádu s přídavným dopravním zpožděním, který je alternativně popsán, v závislosti na parametru `itf`, následujícími přenosovými funkcemi:

$$\begin{aligned} \text{itf} = 1: \quad P(s) &= \frac{\text{pb1} \cdot s + \text{pb0}}{s^2 + \text{pa1} \cdot s + \text{pa0}} \cdot e^{-\text{del} \cdot s} \\ \text{itf} = 2: \quad P(s) &= \frac{k0 (\text{tau} \cdot s + 1)}{(\text{tau1} \cdot s + 1) (\text{tau2} \cdot s + 1)} \cdot e^{-\text{del} \cdot s} \\ \text{itf} = 3: \quad P(s) &= \frac{k0 \cdot \text{om}^2 \cdot (\text{tau}/\text{om} \cdot s + 1)}{(s^2 + 2 \cdot \text{xi} \cdot \text{om} \cdot s + \text{om}^2)} \cdot e^{-\text{del} \cdot s} \\ \text{itf} = 4: \quad P(s) &= \frac{k0 (\text{tau} \cdot s + 1)}{(\text{tau1} \cdot s + 1) s} \cdot e^{-\text{del} \cdot s} \end{aligned}$$

Pro simulaci systémů prvního řádu s dopravním zpožděním (FOPDT) použijte blok [LLC](#) s parametrem `a` nastaveným na nulu.

Diskretní simulace používá přesnou diskretizaci přenosu $P(s)$ pro periodu T_S , s níž je blok SOPDT spouštěn.

Tento blok propaguje kvalitu signálu. Více informací je uvedeno v sekci [1.4](#).

Vstup

`u` Analogový vstupní signál Double (F64)

Parametr

<code>itf</code>	Tvar přenosové funkce	⊙1 Long (I32)
	1 obecný tvar	
	2 reálné póly	
	3 komplexní póly	
	4 s integrátorem	
<code>k0</code>	Statické zesílení	⊙1.0 Double (F64)
<code>tau</code>	Časová konstanta v čitateli	Double (F64)
<code>tau1</code>	První časová konstanta	⊙1.0 Double (F64)

<code>tau2</code>	Druhá časová konstanta	$\odot 1.0$	Double (F64)
<code>om</code>	Vlastní frekvence	$\odot 1.0$	Double (F64)
<code>xi</code>	Součinitel relativního tlumení	$\odot 1.0$	Double (F64)
<code>pb0</code>	Koeficient čitatele: s^0	$\odot 1.0$	Double (F64)
<code>pb1</code>	Koeficient čitatele: s^1	$\odot 1.0$	Double (F64)
<code>pa0</code>	Koeficient jmenovatele: s^0	$\odot 1.0$	Double (F64)
<code>pa1</code>	Koeficient jmenovatele: s^1	$\odot 1.0$	Double (F64)
<code>del</code>	Dopravní zpoždění [s]		Double (F64)
<code>nmax</code>	Rezervovaná paměť pro pole	$\downarrow 10 \uparrow 100000000 \odot 1000$	Long (I32)

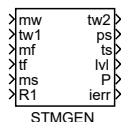
Výstup

<code>y</code>	Analogový výstupní signál	Double (F64)

STMGEN – Model parogenerátoru

Symbol bloku

Licence: [MODEL](#)



Popis funkce

Popis tohoto bloku ještě není k dispozici. Níže naleznete částečný popis vstupů, výstupů a parametrů bloku. Kompletní popis bloku bude k dispozici v dalších revizích dokumentace.

Tento blok nepropaguje kvalitu signálu. Více informací je uvedeno v sekci [1.4](#).

Vstup

mw	Průtok ohřívací vody z reaktoru [kg/s]	Double (F64)
tw1	Teplota ohřívací vody z reaktoru [°C]	Double (F64)
mf	Průtok napájecí vody [kg/s]	Double (F64)
tf	Teplota napájecí vody [°C]	Double (F64)
ms	Průtok vystupující páry [kg/s]	Double (F64)
R1	reset modelu	Bool

Parametr

dsg	Vnitřní průměr parogenerátoru [m]	↓0.0 ⊕4.0	Double (F64)
lsg	Délka trubek parogenerátoru [m]	↓0.0 ⊕13.2	Double (F64)
npipe	Celkový počet trubek	↓0 ⊕8000	Long (I32)
nseg	Počet segmentů (řad trubek)	↓0 ⊕10	Long (I32)
dpipe	Vnější průměr trubek [m]	↓0.0 ⊕0.04	Double (F64)
thpipe	Tloušťka stěn trubek [m]	↓0.0 ⊕0.005	Double (F64)
hl	Vzdálenost spodní řady trubek ode dna [m]	↓0.0 ⊕0.5	Double (F64)
hh	Vzdálenost horní řady trubek ode dna [m]	↓0.0 ⊕3.5	Double (F64)
vExt	Objem Hlavního Parního Kolektoru [m ³]	↓0.0	Double (F64)
ww	Průměr vodní komory/kolektoru [m]	↓0.0 ⊕0.4	Double (F64)
kq	Koeficient prostupu tepla [kW/K/m ²]	↓0.0 ⊕3.2	Double (F64)
t0	Počáteční teplota [°C]	↓0.0 ⊕280.0	Double (F64)
lv10	Počáteční výška hladiny (ode dna) [m]	↓0.0	Double (F64)

Výstup

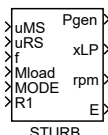
tw2	Teplota vody zpět do reaktoru [°C]	Double (F64)

ps	Výstupní tlak páry [MPa]	Double (F64)
ts	Výstupní teplota páry [°C]	Double (F64)
lvl	Hladina napájecí vody ode dna [m]	Double (F64)
P	Aktuální tepelný výkon [kW]	Double (F64)
ierr	Kód chyby	Error

STURB – Model parní turbíny

Symbol bloku

Licence: [MODEL](#)



Popis funkce

Popis tohoto bloku ještě není k dispozici. Níže naleznete částečný popis vstupů, výstupů a parametrů bloku. Kompletní popis bloku bude k dispozici v dalších revizích dokumentace.

Tento blok nepropaguje kvalitu signálu. Více informací je uvedeno v sekci [1.4](#).

Vstup

uMS	Požadovaná hodnota otevření hlavního ventilu[0..1]	Double (F64)
uRS	Požadovaná hodnota otevření přehřívacího ventilu[0..1]	Double (F64)
f	Frekvence sítě [Hz]	Double (F64)
Mload	Zátěžný moment [Nm]	Double (F64)
MODE	1=generator přifázovaný k síti, 0=ostrovní režim	Bool
R1	Reset modelu	Bool

Parametr

Kms	Časová konstanta hlavního ventilu [1/s]	⊙1.0	Double (F64)
Khp	Časová konstanta vysokotlakého dílu [1/s]	⊙1.0	Double (F64)
Krs	Časová konstanta přehřívacího ventilu [1/s]	⊙1.0	Double (F64)
Krh	Časová konstanta přehříváku [1/s]	⊙1.0	Double (F64)
Kip	Časová konstanta středotlakého dílu [1/s]	⊙1.0	Double (F64)
Klp	Časová konstanta nízkotlakého dílu [1/s]	⊙1.0	Double (F64)
p	Tlak páry [bar]	⊙1.0	Double (F64)
J	Moment setrvačnosti [kg.m ²]	⊙1.0	Double (F64)
dhhp	Rozdíl entalpie na vysokotlakém dílu [J/kg]	⊙1.0	Double (F64)
dhip	Rozdíl entalpie na středotlakém dílu [J/kg]	⊙1.0	Double (F64)
dhlp	Rozdíl entalpie na nízkotlakém dílu [J/kg]	⊙1.0	Double (F64)
B	Koeficient tření [Nm.s/rad]	⊙1.0	Double (F64)
Mfmax	Maximální točivý moment [Nm]	⊙1.0	Double (F64)
K1	Koeficient modelu zátěžového momentu [Nm]	⊙1.0	Double (F64)
K2	Koeficient modelu zátěžového momentu [Nm.s]	⊙1.0	Double (F64)
Pmin	hraniční (minimální) výkon [W]	⊙1.0	Double (F64)

hMS_0	Počáteční hodnota polohy hlavního ventilu [0..1]	⊙1.0	Double (F64)
xHP_0	Počáteční hodnota průtoku páry v turbíně (vysokotlaký díl) [kg/s]	⊙1.0	Double (F64)
hRS_0	Počáteční hodnota polohy přehřívacího ventilu [0..1]	⊙1.0	Double (F64)
xRH_0	Počáteční hodnota průtoku páry v přehříváku [kg/s]	⊙1.0	Double (F64)
xIP_0	Počáteční hodnota průtoku páry v turbíně (středotlaký díl) [kg/s]	⊙1.0	Double (F64)
xLP_0	Počáteční hodnota průtoku páry v turbíně (nizkotlaký díl) [kg/s]	⊙1.0	Double (F64)
e_0	Počáteční hodnota skluzového úhlu [rad]	⊙1.0	Double (F64)
n_0	Počáteční hodnota otáček [Hz]	⊙1.0	Double (F64)

Výstup

Pgen	Výstupní výkon[W]	Double (F64)
xLP	Výstupní hmotností průtok páry [kg/s]	Double (F64)
rpm	Otáčky za minutu	Double (F64)
E	Příznak chyby	Bool

Kapitola 14

MATRIX – Bloky pro maticové a vektorové operace

Obsah

CNA – Konstantní pole (vektor/matice)	460
MB_DASUM – Součet absolutních hodnot	461
MB_DAXPY – Provádí $y := a*x + y$ pro vektory x, y	463
MB_DCOPY – Kopíruje vektor x do vektoru y	465
MB_DDOT – Skalární součin dvou vektorů	467
MB_DGEMM – Provádí $C := \alpha * \text{op}(A) * \text{op}(B) + \beta * C$, where $\text{op}(X) = X$ or $\text{op}(X) = X^T$	469
MB_DGEMV – Provádí $y := \alpha * A * x + \beta * y$ or $y := \alpha * A^T * x$ $+ \beta * y$	471
MB_DGER – Provádí $A := \alpha * x * y^T + A$	473
MB_DNRM2 – Eukleidovská norma vektoru	475
MB_DROT – Rovinná rotace vektoru	477
MB_DSCAL – Násobení vektoru konstantou	479
MB_DSWAP – Záměna dvou vektorů	481
MB_DTRMM – Provádí $B := \alpha * \text{op}(A) * B$ or $B := \alpha * B * \text{op}(A)$, where $\text{op}(X) = X$ or $\text{op}(X) = X^T$ pro trojúhelníkovou matici A	483
MB_DTRMV – Provádí $x := A * x$ or $x := A^T * x$ pro trojúhelníkovou matici A	485
MB_DTRSV – Řeší jednu ze soustav rovnic $A * x = B$ nebo $A^T * x$ $= B$ pro trojúhelníkovou matici A	487
ML_DGEBAK – Zpětná transformace k ML_DGEBAL levých nebo pravých vlastních vektorů	489
ML_DGEBAL – Vyvážení obecné reálné matice	492
ML_DGEBRD – Redukce obecné reálné matice do bidiagonální formy pomocí ortogonální transformace	494

ML_DGECOM – Odhad převrácené hodnoty čísla podmíněnosti obecné reálné matice	496
ML_DGEES – Výpočet vlastních čísel, Schurovy formy a volitelně matice Schurových vektorů	498
ML_DGEEV – Výpočet vlastních čísel a volitelně levých a/nebo pravých vlastních vektorů	500
ML_DGEHRD – Redukce reálné obecné matice A na horní Hessenbergovu formu	502
ML_DGELQF – Výpočet LQ faktorizace reálné matice A s rozměry M x N	505
ML_DGELSD – Výpočet řešení s minimální normou reálné lineární úlohy nejmenších čtverců	507
ML_DGEQRF – Výpočet QR faktorizace reálné matice A s rozměry M x N	509
ML_DGESDD – Výpočet singulární dekompozice (SVD) reálné matice A s rozměry M x N	511
ML_DLACPY – Kopíruje celou nebo část matice do jiné matice . . .	513
ML_DLANGE – Výpočet některé z maticových norem obecné matice	515
ML_DLASET – Inicializuje mimodiagonální a diagonální prvky matice na zadané hodnoty	517
ML_DTRSYL – Řešení reálné Sylvesterovy rovnice pro kvazitrojúhelníkové matice A a B	519
MX_AT – Hodnota prvku matice/vektoru	521
MX_ATSET – Nastavení hodnoty prvku matice/vektoru	522
MX_CNADD – Přičte skalár ke každému prvku matice/vektoru . . .	523
MX_CNMUL – Vynásobí matici/vektor skalárem	524
MX_CTODPA – Diskretizace spojitého modelu (A,B) do (Ad,Bd) s využitím Padéových aproximací	525
MX_DIM – Dimenze matice/vektoru	527
MX_DIMSET – Nastavení dimenze matice/vektoru	528
MX_DSAGET – Uložení submatice A do matice B	530
MX_DSAREF – Nastavení odkazu na submatici A do matice B . . .	532
MX_DSASET – Uložení matice A do submatice v B	534
MX_DTRNSP – Transpozice obecné matice: $B := \alpha * A^T$	536
MX_DTRNSQ – Transpozice čtvercové matice na místě: $A := \alpha * A^T$	538
MX_FILL – Vyplnění reálné matice/vektoru	540
MX_FNX – Maticové a vektorové skalární funkce	541
MX_MAT – Blok pro uložení dat matice	543
MX_RAND – Náhodně vygenerovaná matice nebo vektor	544
MX_REFCOPY – Kopírování vstupních odkazů na matice A a B do jejich výstupních odkazů	546

MX_SLFS – Ukládání a čtení matice/vektoru do souboru nebo textového řetězce	547
MX_VEC – Blok pro uložení dat vektoru	550
MX_WRITE – Výpis matice/vektoru do konzole/systémového logu	551
RTOV – Vektorový multiplexer	553
SWVMR – Přepínač vektorového/maticového/odkazovacího signálu	555
VTOR – Vektorový demultiplexer	556

Knihovna MATRIX je navržena pro pokročilé maticové výpočty a manipulace. Zahrnuje širokou škálu bloků, jako je **MB_DGEMM**, **MB_DTRMM** a **MB_DGER** pro operace typu matice-matice a matice-vektor. Knihovna obsahuje funkce pro rozklad matic (**ML_DGEBRD**, **ML_DGEQRF**), problémy s vlastními hodnotami (**ML_DGEEV**, **ML_DGEES**) a rozklad singulární hodnoty (**ML_DGESDD**). Kromě toho nabízí pomocné bloky jako **MX_MAT**, **MX_VEC** a **MX_FILL** pro vytváření matic a manipulaci, stejně jako specializované bloky, jako je např. **MX_DTRNSP** pro maticovou transpozici a **MX_RAND** pro generování náhodných matic. Tato knihovna je nezbytná pro složité matematické operace zahrnující matice v různých aplikacích.

Poznámky k implementaci

První prvek matice má index(0,0), první prvek vektoru index (0).

Vektor není samostatný objekt, ale matice s jedním sloupcem. Může být i matice s jedním řádkem, které se říká řádkový vektor. Pokud se někdy v systému REXYGEN vyžaduje vektor, myslí se tím sloupcový (tj. matice s jedním sloupcem).

Matice jsou mezi bloky předávány odkazem. V rámci nějakého bloku (nejčastěji **MX_MAT** nebo **MX_VEC**) se alokuje paměť pro celou matici a pak se jen předává jedno číslo, které znamená adresu této paměti. To znamená, že všechny bloky, které si matici předávají pořád pracují s jednou kopií a vzájemně si matici přepisují. Pokud je potřeba vytvořit kopii matice, je potřeba použít další blok **MX_MAT** a data zkopírovat (např. blokem **MB_DCOPY**).

Některé bloky pracující s vektorem (např. **MB_DCPY**, **RTOV**, **VTOR**) nekontrolují přesné rozměry (takže např. matici 10x10 chápou jako 100 prvkový vektor). Pro tento případ je důležité vědět, že se hodnoty ukládají po sloupcích (tj. prvek (1,0) má index 1, prvek (2,0) má index 2, prvek (0,1) má index 10, prvek (0,2) má index 11, atd.). Tyto bloky nelze použít s podmaticí vrácenou blokem **MX_DSAREF**. Chování v tomto případě je nedefinované.

Většina bloků má odkaz/referenci na matici jak na vstupu tak na výstupu. Obojí je odkaz na stejnou matici, takže je jedno, kam další blok připojíme. Protože ale pořadí spouštění bloků se řídí tokem signálu, připojením na výstup jiného bloku definujeme pořadí spouštění bloků a tím i pořadí vykonávání maticových operací.

CNA – Konstantní pole (vektor/matice)

Symbol bloku

Licence: [STANDARD](#)

Popis funkce

Blok CNA alokuje paměť pro `nmax` prvků typu `etype` vektoru/matice referencovaného výstupem `vec` a inicializuje všechny prvky daty uloženými v parametru `acn`.

Pokud není prázdný řetězcový parametr `filename`, načte inicializační data ze souboru `filename` na hostitelském počítači ve formátu CSV. Oddělovač sloupců může být čárka nebo středník nebo mezera (ale stejný v celém souboru), desetinný oddělovač musí být tečka, oddělovač řádků je nový řádek. Prázdné řádky jsou přeskočeny.

Pokud je parametr `TRN = on`, pak výstupní reference `vec` obsahuje transponovaná data.

Poznámka: V případě `etype = Large (I64)` jsou hodnoty načtené z parametru `acn` převedeny na desetinné číslo s dvojitou přesností kvůli implementačním důvodům, takže můžete ztratit přesnost pro velmi velké hodnoty. Pokud by to mohlo být problém, použijte externí soubor pro inicializaci, který tento problém nemá.

Tento blok propaguje kvalitu signálu. Více informací je uvedeno v sekci [1.4](#).

Parametr

<code>filename</code>	Datový soubor s hodnotami oddělenými čárkou	String
<code>TRN</code>	Transponuj načtenou matici	Bool
<code>nmax</code>	Rezervovaná paměť pro výstupní matici (celkový počet prvků) ↓2 ↑100000000 ⊙100	Long (I32)
<code>etype</code>	Typ prvků 1 Bool 2 Byte (U8) 3 Short (I16) 4 Long (I32) 5 Word (U16) 6 DWord (U32) 7 Float (F32) 8 Double (F64) 10 Large (I64)	⊙8 Long (I32)
<code>acn</code>	Počáteční hodnota pole	⊙[0 1 2 3] Double (F64)

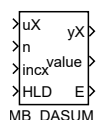
Výstup

<code>vec</code>	Odkaz na vektor/matici dat	Reference

MB_DASUM – Součet absolutních hodnot

Symbol bloku

Licence: [STANDARD](#)



Popis funkce

Výstupní reference yX je vždy nastavena na vstupní referenci uX . Pokud $HLD = on$, pak se nic nepočítá, jinak je interně zavolána funkce BLAS `DASUM`:

```
value = DASUM(N, uX, INCX);
```

kde hodnoty N a $INCX$ jsou nastaveny následovně:

- Pokud je vstup $n > 0$, pak je N nastaveno na n , jinak je N nastaveno na aktuální počet prvků vektoru nebo matice **CNT** referencované vstupem uX .
- Pokud je vstup $incx > 0$, pak je $INCX$ nastaveno na $incx$, jinak je $INCX$ nastaveno na 1.

Chybový příznak E je nastaven na **on**, pokud:

- reference uX není definována (tj. vstup uX není propojen),
- $n < 0$ nebo $incx < 0$,
- $(N - 1) * INCX + 1 > CNT$.

Více informací naleznete v dokumentaci knihovny BLAS [\[10\]](#) (pouze anglicky).

Tento blok nepropaguje kvalitu signálu. Více informací je uvedeno v sekci [1.4](#).

Vstup

uX	Vstupní odkaz na vektor x	Reference
n	Počet zpracovaných prvků vektoru	Long (I32)
$incx$	Přírůstek indexu vektoru x	Long (I32)
HLD	Pozastavení	Bool

Výstup

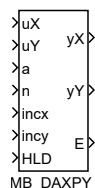
yX	Výstupní odkaz na vektor x	Reference

<code>value</code>	Návratová hodnota funkce	<code>Double (F64)</code>
<code>E</code>	Příznak chyby	<code>Bool</code>

MB_DAXPY – Provádí $y := a * x + y$ pro vektory x, y

Symbol bloku

Licence: [STANDARD](#)



Popis funkce

Výstupní reference yX je vždy nastavena na vstupní referenci uX . Pokud $HLD = on$, pak se nic nepočítá, jinak je interně zavolána funkce BLAS DAXPY:

`DAXPY(N, a, uX, INCX, uY, INCY);`

kde hodnoty N , $INCX$ a $INCY$ jsou nastaveny následovně:

- Pokud je vstup $n > 0$, pak je N nastaveno na n , jinak je N nastaveno na aktuální počet prvků vektoru nebo matice $CNTY$ referencované vstupem uY .
- Pokud je vstup $incx \neq 0$, pak je $INCX$ nastaveno na $incx$, jinak je $INCX$ nastaveno na 1.
- Pokud je vstup $incy \neq 0$ pak je $INCY$ nastaveno na $incy$, jinak je $INCY$ nastaveno na 1.

Chybový příznak E je nastaven na `on`, pokud:

- reference uX nebo uY není definována (tj. vstup uX nebo uY není propojen),
- $n < 0$,
- $(N - 1) * |INCX| + 1 > CNTX$, kde $CNTX$ je počet prvků vektoru nebo matice referencované vstupem uX ,
- $(N - 1) * |INCY| + 1 > CNTY$.

Více informací naleznete v dokumentaci knihovny BLAS [\[10\]](#) (pouze anglicky).

Tento blok nepropaguje kvalitu signálu. Více informací je uvedeno v sekci [1.4](#).

Vstup

uX

Vstupní odkaz na vektor x

Reference

uY	Vstupní odkaz na vektor y	Reference
a	Skalární koeficient a	Double (F64)
n	Počet zpracovaných prvků vektoru	Long (I32)
incx	Přírůstek indexu vektoru x	Long (I32)
incy	Přírůstek indexu vektoru y	Long (I32)
HLD	Pozastavení	Bool

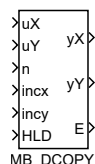
Výstup

yX	Výstupní odkaz na vektor x	Reference
yY	Výstupní odkaz na vektor y	Reference
E	Příznak chyby	Bool

MB_DCOPY – Kopíruje vektor x do vektoru y

Symbol bloku

Licence: [STANDARD](#)



Popis funkce

Výstupní reference yX a yY jsou vždy nastaveny na odpovídající vstupní reference uX a uY . Pokud $HLD = on$, pak se nic nepočítá, jinak je interně zavolána funkce BLAS DCOPY:

`DCOPY(N, uX, INCX, uY, INCY);`

kde hodnoty N , $INCX$ a $INCY$ jsou nastaveny následovně:

- Pokud je vstup $n > 0$, pak je N nastaveno na n , jinak je N nastaveno na aktuální počet prvků vektoru nebo matice $CNTY$ referencované vstupem uY .
- Pokud je vstup $incx \neq 0$, pak je $INCX$ nastaveno na $incx$, jinak je $INCX$ nastaveno na 1.
- Pokud je vstup $incy \neq 0$ pak je $INCY$ nastaveno na $incy$, jinak je $INCY$ nastaveno na 1.

Chybový příznak E je nastaven na on , pokud:

- reference uX nebo uY není definována (tj. vstup uX nebo uY není propojen),
- $n < 0$,
- $(N - 1) * |INCX| + 1 > CNTX$,
- $(N - 1) * |INCY| + 1 > CNTY$, kde $CNTY$ je počet prvků vektoru nebo matice referencované vstupem uY .

Více informací naleznete v dokumentaci knihovny BLAS [\[10\]](#) (pouze anglicky).

Tento blok nepropaguje kvalitu signálu. Více informací je uvedeno v sekci [1.4](#).

Vstup

uX	Vstupní odkaz na vektor x	Reference
uY	Vstupní odkaz na vektor y	Reference

n	Počet zpracovaných prvků vektoru	Long (I32)
incx	Přírůstek indexu vektoru x	Long (I32)
incy	Přírůstek indexu vektoru y	Long (I32)
HLD	Pozastavení	Bool

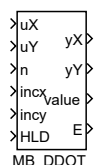
Výstup

yX	Výstupní odkaz na vektor x	Reference
yY	Výstupní odkaz na vektor y	Reference
E	Příznak chyby	Bool

MB_DDOT – Skalární součin dvou vektorů

Symbol bloku

Licence: [STANDARD](#)



Popis funkce

Výstupní reference yX a yY jsou vždy nastaveny na odpovídající vstupní reference uX a uY . Pokud $HLD = on$, pak se nic nepočítá, jinak je interně zavolána funkce BLAS DDOT:

`DDOT(N, uX, INCX, uY, INCY);`

kde hodnoty N , $INCX$ a $INCY$ jsou nastaveny následovně:

- Pokud je vstup $n > 0$, pak je N nastaveno na n , jinak je N nastaveno na aktuální počet prvků vektoru nebo matice $CNTX$ referencované vstupem uX .
- Pokud je vstup $incx \neq 0$, pak je $INCX$ nastaveno na $incx$, jinak je $INCX$ nastaveno na 1.
- Pokud je vstup $incy \neq 0$ pak je $INCY$ nastaveno na $incy$, jinak je $INCY$ nastaveno na 1.

Chybový příznak E je nastaven na **on**, pokud:

- reference uX nebo uY není definována (tj. vstup uX nebo uY není propojen),
- $n < 0$,
- $(N - 1) * |INCX| + 1 > CNTX$,
- $(N - 1) * |INCY| + 1 > CNTY$, kde $CNTY$ je počet prvků vektoru nebo matice referencované vstupem uY .

Více informací naleznete v dokumentaci knihovny BLAS [\[10\]](#) (pouze anglicky).

Tento blok nepropaguje kvalitu signálu. Více informací je uvedeno v sekci [1.4](#).

Vstup

uX	Vstupní odkaz na vektor x	Reference
uY	Vstupní odkaz na vektor y	Reference

468 KAPITOLA 14. MATRIX – BLOKY PRO MATICOVÉ A VEKTOROVÉ OPERACE

<code>n</code>	Počet zpracovaných prvků vektoru	Long (I32)
<code>incx</code>	Přírůstek indexu vektoru x	Long (I32)
<code>incy</code>	Přírůstek indexu vektoru y	Long (I32)
<code>HLD</code>	Pozastavení	Bool

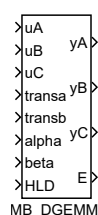
Výstup

<code>yX</code>	Výstupní odkaz na vektor x	Reference
<code>yY</code>	Výstupní odkaz na vektor y	Reference
<code>value</code>	Návratová hodnota funkce	Double (F64)
<code>E</code>	Příznak chyby	Bool

MB_DGEMM – Provádí $C := \alpha * \text{op}(A) * \text{op}(B) + \beta * C$, where $\text{op}(X) = X$ or $\text{op}(X) = X^T$

Symbol bloku

Licence: [STANDARD](#)



Popis funkce

Výstupní reference **yA**, **yB** a **yC** jsou vždy nastaveny na odpovídající vstupní reference **uA**, **uB** a **uC**. Pokud **HLD** = **on**, pak se nic nepočítá, jinak je interně zavolána funkce BLAS **DGEMM**:

```
DGEMM(sTRANSA, sTRANSB, M, N, KA, alpha, uA, LDA, uB, LDB, beta, uC, LDC);
```

kde parametry funkce **DGEMM** jsou nastaveny následovně:

- Celočíselné vstupy **transa** a **transb** jsou mapovány na řetězce **sTRANSA** a **sTRANSB**: $\{0, 1\} \rightarrow \text{"N"}, \{2\} \rightarrow \text{"T"} \text{ a } \{3\} \rightarrow \text{"C"}$.
- **M** je počet řádků matice referencované vstupem **uC**.
- **N** je počet sloupců matice referencované vstupem **uC**.
- Pokud je vstup **transa** roven 0 nebo 1, pak je **KA** počet sloupců, jinak je **KA** počet řádků matice referencované vstupem **uA**.
- **LDA**, **LDB** a **LDC** jsou vedoucí rozměry matic referencovaných vstupy **uA**, **uB** a **uC**.

Chybový příznak **E** je nastaven na **on**, pokud:

- reference **uA** nebo **uB** nebo **uC** není definována (tj. vstup **uA** nebo **uB** nebo **uC** není propojen),
- **transa** nebo **transb** je menší než 0 nebo větší než 3,
- $KA \neq KB$; pokud je vstup **transb** roven 0 nebo 1, pak je **KB** počet řádků, jinak je **KB** počet sloupců matice referencované vstupem **uB** (tj. matice $\text{op}(A)$ a $\text{op}(B)$ musí být násobitelné).
- volání funkce **DGEMM** vrátí chybu pomocí funkce **XERBLA**, viz systémový log.

Více informací naleznete v dokumentaci knihovny BLAS [\[10\]](#) (pouze anglicky).

Tento blok nepropaguje kvalitu signálu. Více informací je uvedeno v sekci [1.4](#).

Vstup

uA	Vstupní odkaz na matici A		Reference
uB	Vstupní odkaz na matici B		Reference
uC	Vstupní odkaz na matici C		Reference
transa	Transpozice matice A	↓0 ↑3	Long (I32)
transb	Transpozice matice B	↓0 ↑3	Long (I32)
alpha	Skalární koeficient alpha		Double (F64)
beta	Skalární koeficient beta		Double (F64)
HLD	Pozastavení		Bool

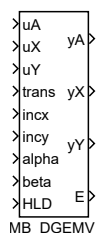
Výstup

yA	Výstupní odkaz na matici A	Reference
yB	Výstupní odkaz na matici B	Reference
yC	Výstupní odkaz na matici C	Reference
E	Příznak chyby	Bool

MB_DGEMV – Provádí $y := \alpha * A * x + \beta * y$ or $y := \alpha * A^T * x + \beta * y$

Symbol bloku

Licence: [STANDARD](#)



Popis funkce

Výstupní reference **yA**, **yX** a **yY** jsou vždy nastaveny na odpovídající vstupní reference **uA**, **uX** a **uY**. Pokud **HLD = on**, pak se nic nepočítá, jinak je interně zavolána funkce BLAS **DGEMM**:

DGEMV(sTRANS, M, N, alpha, uA, LDA, uX, INCX, beta, uY, INCY);

kde parametry funkce **DGEMV** jsou nastaveny následovně:

- Celočíselný vstup **trans** je mapován na řetězec **sTRANS**: $\{0, 1\} \rightarrow "N"$, $\{2\} \rightarrow "T"$ a $\{3\} \rightarrow "C"$.
- **M** je počet řádků matice referencované vstupem **uA**.
- **N** je počet sloupců matice referencované vstupem **uA**.
- **LDA** je vedoucí rozměr matice referencované vstupem **uA**.
- Pokud je vstup **incx** $\neq 0$, pak je **INCX** nastaveno na **incx**, jinak je **INCX** nastaveno na 1.
- Pokud je vstup **incy** $\neq 0$, pak je **INCY** nastaveno na **incy**, jinak je **INCY** nastaveno na 1.

Chybový příznak **E** je nastaven na **on**, pokud:

- reference **uA** nebo **uX** nebo **uY** není definována (tj. vstup **uA** nebo **uX** nebo **uY** není propojen),
- **trans** je menší než 0 nebo větší než 3,
- volání funkce **DGEMV** vrátí chybu pomocí funkce **XERBLA**, viz systémový log.

Více informací naleznete v dokumentaci knihovny BLAS [\[10\]](#) (pouze anglicky).

Tento blok nepropaguje kvalitu signálu. Více informací je uvedeno v sekci [1.4](#).

Vstup

<code>uA</code>	Vstupní odkaz na matici A		Reference
<code>uX</code>	Vstupní odkaz na vektor x		Reference
<code>uY</code>	Vstupní odkaz na vektor y		Reference
<code>trans</code>	Transpozice vstupní matice	↓0 ↑3	Long (I32)
<code>incx</code>	Přírůstek indexu vektoru x		Long (I32)
<code>incy</code>	Přírůstek indexu vektoru y		Long (I32)
<code>alpha</code>	Skalární koeficient alpha		Double (F64)
<code>beta</code>	Skalární koeficient beta		Double (F64)
<code>HLD</code>	Pozastavení		Bool

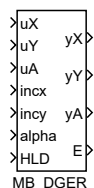
Výstup

<code>yA</code>	Výstupní odkaz na matici A	Reference
<code>yX</code>	Výstupní odkaz na vektor x	Reference
<code>yY</code>	Výstupní odkaz na vektor y	Reference
<code>E</code>	Příznak chyby	Bool

MB_DGER – Provádí $A := \alpha * x * y^T + A$

Symbol bloku

Licence: [STANDARD](#)



Popis funkce

Výstupní reference yX , yY a yA jsou vždy nastaveny na odpovídající vstupní reference uX , uY a uA . Pokud $HLD = \text{on}$, pak se nic nepočítá, jinak je interně zavolána funkce BLAS DGER:

`DGER(M, N, alpha, uX, INCX, uY, INCY, uA, LDA);`

kde parametry funkce `DGER` jsou nastaveny následovně:

- M je počet řádků matice referencované vstupem uA .
- N je počet sloupců matice referencované vstupem uA .
- Pokud je vstup $incx \neq 0$, pak je $INCX$ nastaveno na $incx$, jinak je $INCX$ nastaveno na 1.
- Pokud je vstup $incy \neq 0$, pak je $INCY$ nastaveno na $incy$, jinak je $INCY$ nastaveno na 1.
- LDA je vedoucí rozměr matice referencované vstupem uA .

Chybový příznak E je nastaven na `on`, pokud:

- reference uX nebo uY nebo uA není definována (tj. vstup uX nebo uY nebo uA není propojen),
- volání funkce `DGER` vrátí chybu pomocí funkce `XERBLA`, viz systémový log.

Více informací naleznete v dokumentaci knihovny BLAS [\[10\]](#) (pouze anglicky).

Tento blok nepropaguje kvalitu signálu. Více informací je uvedeno v sekci [1.4](#).

Vstup

uX

Vstupní odkaz na vektor x

Reference

uY	Vstupní odkaz na vektor y	Reference
uA	Vstupní odkaz na matici A	Reference
incx	Přírůstek indexu vektoru x	Long (I32)
incy	Přírůstek indexu vektoru y	Long (I32)
alpha	Skalární koeficient alpha	Double (F64)
HLD	Pozastavení	Bool

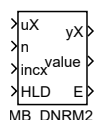
Výstup

yX	Výstupní odkaz na vektor x	Reference
yY	Výstupní odkaz na vektor y	Reference
yA	Výstupní odkaz na matici A	Reference
E	Příznak chyby	Bool

MB_DNRM2 – Eukleidovská norma vektoru

Symbol bloku

Licence: [STANDARD](#)



Popis funkce

Výstupní reference `yX` je vždy nastavena na vstupní referenci `uX`. Pokud je `HLD = on`, pak se nic nepočítá, jinak je interně zavolána funkce BLAS `DNRM2`:

```
value = DNRM2(N, uX, INCX);
```

kde hodnoty `N` a `INCX` jsou nastaveny následovně:

- Pokud je vstup `n > 0`, pak je `N` nastaveno na `n`, jinak je `N` nastaveno na aktuální počet prvků vektoru nebo matice `CNT` referencovaných vstupem `uX`.
- Pokud je vstup `incx > 0`, pak je `INCX` nastaveno na `incx`, jinak je `INCX` nastaveno na 1.

Chybový příznak `E` je nastaven na `on`, pokud:

- reference `uX` není definována (tj. vstup `uX` není propojen),
- `n < 0` nebo `incx < 0`,
- $(N - 1) * |INCX| + 1 > CNT$.

Více informací naleznete v dokumentaci knihovny BLAS [\[10\]](#) (pouze anglicky).

Použijte blok [ML_DLANGE](#) pro výpočet různých norm matic.

Tento blok nepropaguje kvalitu signálu. Více informací je uvedeno v sekci [1.4](#).

Vstup

<code>uX</code>	Vstupní odkaz na vektor <code>x</code>	Reference
<code>n</code>	Počet zpracovaných prvků vektoru	Long (I32)
<code>incx</code>	Přírůstek indexu vektoru <code>x</code>	Long (I32)
<code>HLD</code>	Pozastavení	Bool

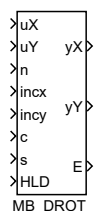
Výstup

<code>yX</code>	Výstupní odkaz na vektor <code>x</code>	Reference
<code>value</code>	Návratová hodnota funkce	Double (F64)
<code>E</code>	Příznak chyby	Bool

MB_DROT – Rovinná rotace vektoru

Symbol bloku

Licence: [STANDARD](#)



Popis funkce

Výstupní reference yX a yY jsou vždy nastaveny na odpovídající vstupní reference uX a uY . Pokud je $HLD = on$, pak se nic nepočítá, jinak je interně zavolána funkce BLAS DROT:

$DROT(N, uX, INCX, uY, INCY, c, s);$

kde parametry funkce DROT jsou nastaveny následovně:

- Pokud je vstup $n > 0$, pak je N nastaveno na n , jinak je N nastaveno na aktuální počet prvků vektoru nebo matice $CNTX$ referencovaných vstupem uX .
- Pokud je vstup $incx \neq 0$, pak je $INCX$ nastaveno na $incx$, jinak je $INCX$ nastaveno na 1.
- Pokud je vstup $incy \neq 0$, pak je $INCY$ nastaveno na $incy$, jinak je $INCY$ nastaveno na 1.

Chybový příznak E je nastaven na on , pokud:

- reference uX nebo uY není definována (tj. vstup uX nebo uY není propojen),
- $n < 0$,
- $(N - 1) * |INCX| + 1 > CNTX$,
- $(N - 1) * |INCY| + 1 > CNTY$, kde $CNTY$ je počet prvků vektoru nebo matice referencovaných vstupem uY .

Více informací naleznete v dokumentaci knihovny BLAS [\[10\]](#) (pouze anglicky).

Tento blok nepropaguje kvalitu signálu. Více informací je uvedeno v sekci [1.4](#).

Vstup

uX	Vstupní odkaz na vektor x	Reference

uY	Vstupní odkaz na vektor y	Reference
n	Počet zpracovaných prvků vektoru	Long (I32)
incx	Přírůstek indexu vektoru x	Long (I32)
incy	Přírůstek indexu vektoru y	Long (I32)
c	Skalární koeficient c	Double (F64)
s	Skalární koeficient s	Double (F64)
HLD	Pozastavení	Bool

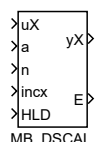
Výstup

yX	Výstupní odkaz na vektor x	Reference
yY	Výstupní odkaz na vektor y	Reference
E	Příznak chyby	Bool

MB_DSCAL – Násobení vektoru konstantou

Symbol bloku

Licence: [STANDARD](#)



Popis funkce

Výstupní reference `yX` je vždy nastavena na odpovídající vstupní reference `uX`. Pokud je `HLD = on`, pak se nic nepočítá, jinak je interně zavolána funkce BLAS `DSCAL`:

```
DSCAL(N, a, uX, INCX);
```

kde parametry funkce `DSCAL` jsou nastaveny následovně:

- Pokud je vstup `n > 0`, pak je `N` nastaveno na `n`, jinak je `N` nastaveno na aktuální počet prvků vektoru nebo matice `CNT` referencovaných vstupem `uX`.
- Pokud je vstup `incx ≠ 0`, pak je `INCX` nastaveno na `incx`, jinak je `INCX` nastaveno na 1.

Chybový příznak `E` je nastaven na `on`, pokud:

- reference `uX` není definována (tj. vstup `uX` není propojen),
- `n < 0` nebo `incx < 0`,
- $(N - 1) * INCX + 1 > CNT$.

Více informací naleznete v dokumentaci knihovny BLAS [\[10\]](#) (pouze anglicky).

Tento blok nepropaguje kvalitu signálu. Více informací je uvedeno v sekci [1.4](#).

Vstup

		Reference
<code>uX</code>	Vstupní odkaz na vektor <code>x</code>	<code>Double (F64)</code>
<code>a</code>	Skalární koeficient <code>a</code>	<code>Long (I32)</code>
<code>n</code>	Počet zpracovaných prvků vektoru	<code>Long (I32)</code>
<code>incx</code>	Přírůstek indexu vektoru <code>x</code>	<code>Long (I32)</code>
<code>HLD</code>	Pozastavení	<code>Bool</code>

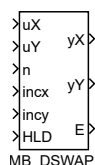
Výstup

yX	Výstupní odkaz na vektor x	Reference
E	Příznak chyby	Bool

MB_DSWAP – Záměna dvou vektorů

Symbol bloku

Licence: [STANDARD](#)



Popis funkce

Výstupní reference yX a yY jsou vždy nastaveny na odpovídající vstupní reference uX a uY . Pokud je $HLD = on$, pak se nic nepočítá, jinak je interně zavolána funkce BLAS DSWAP:

$DSWAP(N, uX, INCX, uY, INCY);$

kde hodnoty N , $INCX$ a $INCY$ jsou nastaveny následovně:

- Pokud je vstup $n > 0$, pak je N nastaveno na n , jinak je N nastaveno na aktuální počet prvků vektoru nebo matice $CNTX$ referencované vstupem uX .
- Pokud je vstup $incx \neq 0$, pak je $INCX$ nastaveno na $incx$, jinak je $INCX$ nastaveno na 1.
- Pokud je vstup $incy \neq 0$, pak je $INCY$ nastaveno na $incy$, jinak je $INCY$ nastaveno na 1.

Chybový příznak E je nastaven na on , pokud:

- reference uX nebo uY není definována (tj. vstup uX nebo uY není připojen),
- $n < 0$,
- $(N - 1) * |INCX| + 1 > CNTX$,
- $(N - 1) * |INCY| + 1 > CNTY$, kde $CNTY$ je počet prvků vektoru nebo matice referencované vstupem uY .

Více informací naleznete v dokumentaci knihovny BLAS [\[10\]](#) (pouze anglicky).

Tento blok nepropaguje kvalitu signálu. Více informací je uvedeno v sekci [1.4](#).

Vstup

uX

Vstupní odkaz na vektor x

Reference

482 KAPITOLA 14. MATRIX – BLOKY PRO MATICOVÉ A VEKTOROVÉ OPERACE

uY	Vstupní odkaz na vektor y	Reference
n	Počet zpracovaných prvků vektoru	Long (I32)
incx	Přírůstek indexu vektoru x	Long (I32)
incy	Přírůstek indexu vektoru y	Long (I32)
HLD	Pozastavení	Bool

Výstup

yX	Výstupní odkaz na vektor x	Reference
yY	Výstupní odkaz na vektor y	Reference
E	Příznak chyby	Bool

MB_DTRMM – Provádí $B := \alpha * \text{op}(A) * B$ or $B := \alpha * B * \text{op}(A)$, where $\text{op}(X) = X$ or $\text{op}(X) = X^T$ pro trojúhelníkovou matici A

Symbol bloku

Licence: [STANDARD](#)



Popis funkce

Výstupní reference yA a yB jsou vždy nastaveny na odpovídající vstupní reference uA a uB . Pokud je $HLD = \text{on}$, pak se nic nepočítá, jinak je interně zavolána funkce BLAS DTRMM:

`DTRMM(sRSIDE, sLUPLO, sTRANSa, sNDIAG, M, N, alpha, uA, LDA, uB, LDB);`

kde parametry funkce DTRMM jsou nastaveny následovně:

- Pokud $RSIDE = \text{on}$, pak je řetězec `sRSIDE` nastaven na "R", jinak je nastaven na "L".
- Pokud $LUPLO = \text{on}$, pak je řetězec `sLUPLO` nastaven na "L", jinak je nastaven na "U".
- Celočíselný vstup `transa` je mapován na řetězec `sTRANSa`: $\{0, 1\} \rightarrow \text{"N"}$, $\{2\} \rightarrow \text{"T"}$ a $\{3\} \rightarrow \text{"C"}$.
- Pokud $NDIAG = \text{on}$, pak je řetězec `sNDIAG` nastaven na "N", jinak je nastaven na "U".
- M je počet řádků matice referencované vstupem `uB`.
- N je počet sloupců matice referencované vstupem `uB`.
- LDA a LDB jsou hlavní rozměry matic referencovaných vstupy `uA` a `uB`.

Chybový příznak E je nastaven na `on`, pokud:

- reference `uA` nebo `uB` není definována (tj. vstup `uA` nebo `uB` není připojen),
- `transa` je menší než 0 nebo větší než 3,

- matice referencovaná vstupem `uA` není čtvercová nebo není kompatibilní s maticí referencovanou vstupem `uB`,
- volání funkce `DTRMM` vrátí chybu pomocí funkce `XERBLA`, viz systémový log.

Více informací naleznete v dokumentaci knihovny BLAS [10] (pouze anglicky).

Tento blok nepropaguje kvalitu signálu. Více informací je uvedeno v sekci 1.4.

Vstup

<code>uA</code>	Vstupní odkaz na matici A	Reference
<code>uB</code>	Vstupní odkaz na matici B	Reference
<code>RSIDE</code>	Operace je aplikována z pravé strany	Bool
<code>LUPL0</code>	Matice A je dolní trojúhelníková matice	Bool
<code>transa</code>	Transpozice matice A	↓0 ↑3 Long (I32)
<code>NDIAG</code>	Nepředpokládá se, že matice A má na diagonále jedničky	Bool
<code>alpha</code>	Skalární koeficient alpha	Double (F64)
<code>HLD</code>	Pozastavení	Bool

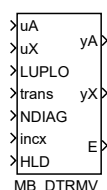
Výstup

<code>yA</code>	Výstupní odkaz na matici A	Reference
<code>yB</code>	Výstupní odkaz na matici B	Reference
<code>E</code>	Příznak chyby	Bool

MB_DTRMV – Provádí $x := A * x$ or $x := A^T * x$ pro trojúhelníkovou matici A

Symbol bloku

Licence: [STANDARD](#)



Popis funkce

Výstupní reference yA a yX jsou vždy nastaveny na odpovídající vstupní reference uA a uX . Pokud je $HLD = on$, pak se nic nepočítá, jinak je interně zavolána funkce BLAS DTRMV:

```
DTRMV(sLUPLO, sTRANS, sNDIAG, N, uA, LDA, uX, INCX);
```

kde jsou parametry funkce DTRMV nastaveny následovně:

- Pokud $LUPLO = on$, pak je řetězec $sLUPLO$ nastaven na "L" jinak je nastaven na "U".
- Celočíselný vstup $trans$ je mapován na řetězec $sTRANS$: $\{0, 1\} \rightarrow "N"$, $\{2\} \rightarrow "T"$ a $\{3\} \rightarrow "C"$.
- Pokud $NDIAG = on$, pak je řetězec $sNDIAG$ nastaven na "N" jinak je nastaven na "U".
- N je počet řádků a sloupců čtvercové matice referencované uA .
- LDA je hlavní rozměr matice referencované uA .
- Pokud vstup $incx \neq 0$ pak je $INCX$ nastaveno na $incx$ jinak je nastaveno na 1.

Chybový příznak E je nastaven na on , pokud:

- reference uA nebo uX není definována (tj. vstup uA nebo uX není připojen),
- $trans$ je menší než 0 nebo větší než 3,
- matice referencovaná uA není čtvercová,
- $(N - 1) * |INCX| + 1 > CNTX$, kde $CNTX$ je počet prvků vektoru nebo matice referencované uX .

- volání funkce `DTRMV` vrátí chybu pomocí funkce `XERBLA`, viz systémový log.

Více informací naleznete v dokumentaci knihovny BLAS [10] (pouze anglicky).

Tento blok nepropaguje kvalitu signálu. Více informací je uvedeno v sekci 1.4.

Vstup

<code>uA</code>	Vstupní odkaz na matici A		Reference
<code>uX</code>	Vstupní odkaz na vektor x		Reference
<code>LUPL0</code>	Matice A je dolní trojúhelníková matice		Bool
<code>trans</code>	Transpozice vstupní matice	↓0 ↑3	Long (I32)
<code>NDIAG</code>	Nepředpokládá se, že matice A má na diagonále jedničky		Bool
<code>incx</code>	Přírůstek indexu vektoru x		Long (I32)
<code>HLD</code>	Pozastavení		Bool

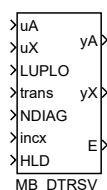
Výstup

<code>yA</code>	Výstupní odkaz na matici A	Reference
<code>yX</code>	Výstupní odkaz na vektor x	Reference
<code>E</code>	Příznak chyby	Bool

MB_DTRSV – Řeší jednu ze soustav rovnic $A * x = B$ nebo $A^T * x = B$ pro trojúhelníkovou matici A

Symbol bloku

Licence: [STANDARD](#)



Popis funkce

Výstupní reference yA a yX jsou vždy nastaveny na odpovídající vstupní reference uA a uX . Pokud je $HLD = on$, pak se nic nepočítá, jinak je interně zavolána funkce BLAS DTRSV:

`DTRSV(sLUPLO, sTRANS, sNDIAG, N, uA, LDA, uX, INCX);`

kde jsou parametry funkce DTRSV nastaveny následovně:

- Pokud $LUPLO = on$, pak je řetězec `sLUPLO` nastaven na "L" jinak je nastaven na "U".
- Celočíselný vstup `trans` je mapován na řetězec `sTRANS`: $\{0, 1\} \rightarrow "N"$, $\{2\} \rightarrow "T"$ a $\{3\} \rightarrow "C"$.
- Pokud $NDIAG = on$, pak je řetězec `sNDIAG` nastaven na "N" jinak je nastaven na "U".
- N je počet řádků a sloupců čtvercové matice referencované uA .
- LDA je hlavní rozměr matice referencované uA .
- Pokud vstup $incx \neq 0$ pak je `INCX` nastaveno na `incx` jinak je nastaveno na 1.

Chybový příznak E je nastaven na `on`, pokud:

- reference uA nebo uX není definována (tj. vstup uA nebo uX není připojen),
- `trans` je menší než 0 nebo větší než 3,
- matice referencovaná uA není čtvercová,
- $(N - 1) * |INCX| + 1 > CNTX$, kde `CNTX` je počet prvků vektoru nebo matice referencované uX .

- volání funkce `DTRSV` vrátí chybu pomocí funkce `XERBLA`, viz systémový log.

Více informací naleznete v dokumentaci knihovny BLAS [\[10\]](#) (pouze anglicky).

Tento blok nepropaguje kvalitu signálu. Více informací je uvedeno v sekci [1.4](#).

Vstup

<code>uA</code>	Vstupní odkaz na matici A		Reference
<code>uX</code>	Vstupní odkaz na vektor x		Reference
<code>LUPL0</code>	Matice A je dolní trojúhelníková matice		Bool
<code>trans</code>	Transpozice vstupní matice	↓0 ↑3	Long (I32)
<code>NDIAG</code>	Nepředpokládá se, že matice A má na diagonále jedničky		Bool
<code>incx</code>	Přírůstek indexu vektoru x		Long (I32)
<code>HLD</code>	Pozastavení		Bool

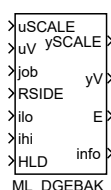
Výstup

<code>yA</code>	Výstupní odkaz na matici A	Reference
<code>yX</code>	Výstupní odkaz na vektor x	Reference
<code>E</code>	Příznak chyby	Bool

ML_DGEBAK – Zpětná transformace k ML_DGEBAL levých nebo pravých vlastních vektorů

Symbol bloku

Licence: [MATRIX](#)



Popis funkce

Výstupní reference `ySCALE` a `yV` jsou vždy nastaveny na odpovídající vstupní reference `uSCALE` a `uV`. Pokud je `HLD = on`, pak se nic nepočítá, jinak je interně zavolána funkce LAPACK `DGEBAK`:

```
DGEBAK(sJOB, sRSIDE, N, ilo, IHI, uSCALE, M, uV, LDV, info);
```

kde jsou parametry funkce `DGEBAK` nastaveny následovně:

- Celočíslný vstup `job` je mapován na řetězec `sJOB`: $\{0, 1\} \rightarrow "N"$, $\{2\} \rightarrow "P"$, $\{3\} \rightarrow "S"$ a $\{4\} \rightarrow "B"$.
- Pokud `RSIDE = on` pak je řetězec `sRSIDE` nastaven na `"R"` jinak je nastaven na `"L"`.
- `N` je počet prvků vektoru referencovaného `uSCALE`.
- Pokud vstup `ihi` $\neq 0$ pak je `IHI` nastaveno na `ihi` jinak je nastaveno na `N - 1`.
- `M` je počet sloupců matice referencované `uV`.
- `LDV` je hlavní rozměr matice referencované `uV`.
- `info` je návratový kód z funkce `DGEBAK`.

Chybový příznak `E` je nastaven na `on`, pokud:

- reference `uSCALE` nebo `uV` není definována (tj. vstup `uSCALE` nebo `uV` není připojen),
- volání funkce `DGEBAK` vrátí chybu pomocí funkce `XERBLA`, viz návratový kód `info` a systémový log.

Upozorňujeme, že indexy `ilo` a `ihi` začínají od nuly na rozdíl od FORTRAN verze, kde začínají od jedničky. Více informací naleznete v dokumentaci knihovny LAPACK [\[11\]](#) (pouze anglicky).

Tento blok nepropaguje kvalitu signálu. Více informací je uvedeno v sekci [1.4](#).

Vstup

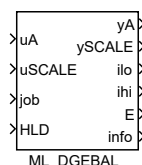
uSCALE	Vstupní odkaz na vektor SCALE	Reference
uV	Odkaz na matici pravých nebo levých vlastních vektorů, které mají být transformovány	Reference
job	Typ požadované zpětné transformace	↓0 ↑4 Long (I32)
RSIDE	Operace je aplikována z pravé strany	Bool
ilo	Dolní index (od nuly) řádku a sloupce pracovní submatice	Long (I32)
ihi	Horní index (od nuly) řádku a sloupce pracovní submatice	Long (I32)
HLD	Pozastavení	Bool

Výstup

ySCALE	Výstupní odkaz na vektor SCALE	Reference
yV	Odkaz na matici transformovaných pravých nebo levých vlastních vektorů	Reference
E	Příznak chyby	Bool
info	Informace o výsledku funkce LAPACKu. Je-li info = -i, pak i-tý argument měl nepřipustnou hodnotu	Long (I32)

ML_DGEBAL – Vyvážení obecné reálné matice

Symbol bloku

Licence: [MATRIX](#)

Popis funkce

Výstupní reference `yA` a `ySCALE` jsou vždy nastaveny na odpovídající vstupní reference `uA` a `uSCALE`. Pokud `HLD = on`, pak není provedeno nic jiného, jinak je interně zavolána funkce LAPACK `DGEBAL`:

```
DGEBAL(sJOB, N, uA, LDA, ilo, ihi, uSCALE, info);
```

kde parametry funkce `DGEBAL` jsou nastaveny následovně:

- Celočíselný vstup `job` je mapován na řetězec `sJOB`: $\{0, 1\} \rightarrow "N"$, $\{2\} \rightarrow "P"$, $\{3\} \rightarrow "S"$ a $\{4\} \rightarrow "B"$.
- `N` je počet sloupců čtvercové matice referencované `uA`.
- `LDA` je hlavní rozměr matice referencované `uA`.
- `ilo` a `ihi` jsou vrácené dolní a horní indexy řádků a sloupců vyvážené submatice matice referencované `uA`.
- `info` je návratový kód z funkce `DGEBAL`.

Chybový příznak `E` je nastaven na `on`, pokud:

- reference `uA` nebo `uSCALE` není definována (tj. vstup `uA` nebo `uSCALE` není připojen),
- matice referencovaná `uA` není čtvercová,
- počet prvků vektoru referencovaného `uSCALE` je menší než `N`,
- volání funkce `DGEBAL` vrátí chybu pomocí funkce `XERBLA`, viz návratový kód `info` a systémový log.

Upozorňujeme, že indexy `ilo` a `ihi` začínají od nuly na rozdíl od verze FORTRAN, kde začínají od jedné. Více informací naleznete v dokumentaci knihovny LAPACK [\[11\]](#) (pouze anglicky).

Tento blok nepropaguje kvalitu signálu. Více informací je uvedeno v sekci [1.4](#).

Vstup

uA	Vstupní odkaz na matici A	Reference
uSCALE	Vstupní odkaz na vektor SCALE	Reference
job	Specifikuje operace, které mají být provedeny s maticí A	↓0 ↑4 Long (I32)
HLD	Pozastavení	Bool

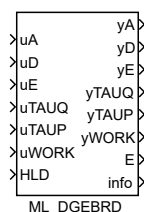
Výstup

yA	Výstupní odkaz na matici A	Reference
ySCALE	Výstupní odkaz na vektor SCALE	Reference
ilo	Dolní index (od nuly) řádku a sloupce pracovní submatice	Long (I32)
ihi	Horní index (od nuly) řádku a sloupce pracovní submatice	Long (I32)
E	Příznak chyby	Bool
info	Informace o výsledku funkce LAPACKu. Je-li info = -i, pak i-tý argument měl nepřipustnou hodnotu	Long (I32)

ML_DGEBRD – Redukce obecné reálné matice do bidiagonální formy pomocí ortogonální transformace

Symbol bloku

Licence: [MATRIX](#)



Popis funkce

Výstupní reference `yA`, `yD`, `yE`, `yTAUQ`, `yTAUP` a `yWORK` jsou vždy nastaveny na odpovídající vstupní reference `uA`, `uD`, `uE`, `uTAUQ`, `uTAUP` a `uWORK`. Pokud `HLD = on`, pak není provedeno nic jiného, jinak je interně zavolána funkce LAPACK `DGEBRD`:

```
DGEBRD(M, N, uA, LDA, uD, uE, uTAUQ, uTAUP, uWORK, info);
```

kde parametry funkce `DGEBRD` jsou nastaveny následovně:

- `M` je počet řádků matice referencované `uA`.
- `N` je počet sloupců matice referencované `uA`.
- `LDA` je hlavní rozměr matice referencované `uA`.
- `info` je návratový kód z funkce `DGEBRD`.

Chybový příznak `E` je nastaven na `on`, pokud:

- reference `uA` nebo `uD` nebo `uE` nebo `uTAUQ` nebo `uTAUP` nebo `uWORK` není definována (tj. vstup `uA` nebo `uD` nebo `uE` nebo `uTAUQ` nebo `uTAUP` nebo `uWORK` není připojen),
- počet prvků vektoru referencovaného `uD`, `uTAUQ` a `uTAUP` je menší než `MINMN`, kde `MINMN` je minimum z `M` a `N`,
- počet prvků vektoru referencovaného `uE` je menší než `MINMN - 1`,
- volání funkce `DGEBRD` vrátí chybu pomocí funkce `XERBLA`, viz návratový kód `info` a systémový log.

Více informací naleznete v dokumentaci knihovny LAPACK [11] (pouze anglicky).

Tento blok nepropaguje kvalitu signálu. Více informací je uvedeno v sekci 1.4.

Vstup

uA	Vstupní odkaz na matici A	Reference
uD	Diagonální prvky bidiagonální matice B	Reference
uE	Mimodiagonální prvky bidiagonální matice B	Reference
uTAUQ	Reference na vektor skalárních faktorů elementárních reflektorů, které reprezentují ortogonální matici Q	Reference
uTAUP	Reference na vektor skalárních faktorů elementárních reflektorů, které reprezentují ortogonální matici P	Reference
uWORK	Vstupní odkaz na pracovní vektor WORK	Reference
HLD	Pozastavení	Bool

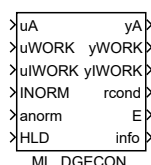
Výstup

yA	Výstupní odkaz na matici A	Reference
yD	Výstupní reference na D	Reference
yE	Výstupní reference na E	Reference
yTAUQ	Výstupní reference na TAUQ	Reference
yTAUP	Výstupní reference na TAUP	Reference
yWORK	Výstupní odkaz na pracovní vektor WORK	Reference
E	Příznak chyby	Bool
info	Informace o výsledku funkce LAPACKu. Je-li info = -i, pak i-tý argument měl nepřipustnou hodnotu	Long (I32)

ML_DGECON – Odhad převrácené hodnoty čísla podmíněnosti obecné reálné matice

Symbol bloku

Licence: [MATRIX](#)



Popis funkce

Výstupní reference `yA`, `yWORK` a `yIWORK` jsou vždy nastaveny na odpovídající vstupní reference `uA`, `uWORK` a `uIWORK`. Pokud `HLD = on`, pak není provedeno nic jiného, jinak je interně zavolána funkce LAPACK `DGECON`:

```
DGECON(sINORM, N, uA, LDA, anorm, rcond, uWORK, uIWORK, info);
```

kde parametry funkce `DGECON` jsou nastaveny následovně:

- Pokud `INORM = on`, pak je řetězec `sINORM` nastaven na "I" jinak je nastaven na "1".
- `N` je počet sloupců matice referencované `uA`.
- `LDA` je hlavní rozměr matice referencované `uA`.
- `rcond` je vrácená reciproční hodnota podmíněnosti matice referencované `uA`.
- `info` je návratový kód z funkce `DGECON`.

Chybový příznak `E` je nastaven na `on`, pokud:

- reference `uA` nebo `uWORK` nebo `uIWORK` není definována (tj. vstup `uA` nebo `uWORK` nebo `uIWORK` není připojen),
- matice referencovaná `uA` není čtvercová,
- počet prvků vektoru referencovaného pomocí `uWORK` je menší než $4 * N$,
- počet prvků celočíselného vektoru referencovaného pomocí `uIWORK` je menší než N ,
- volání funkce `DGECON` vrátí chybu pomocí funkce `XERBLA`, viz návratový kód `info` a systémový log.

Více informací naleznete v dokumentaci knihovny LAPACK [11] (pouze anglicky).

Tento blok nepropaguje kvalitu signálu. Více informací je uvedeno v sekci 1.4.

Vstup

uA	Vstupní odkaz na matici A	Reference
uWORK	Vstupní odkaz na pracovní vektor WORK	Reference
uIWORK	Vstupní odkaz na celočíselný pracovní vektor WORK	Reference
INORM	Použij tzv. Infinity-norm (maximum z řádkových součtů absolutních hodnot prvků)	Bool
anorm	Norma původní matice A	Double (F64)
HLD	Pozastavení	Bool

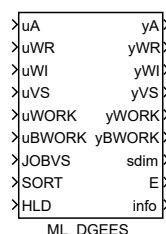
Výstup

yA	Výstupní odkaz na matici A	Reference
yWORK	Výstupní odkaz na pracovní vektor WORK	Reference
yIWORK	Výstupní odkaz na celočíselný pracovní vektor WORK	Reference
rcond	Převrácená hodnota čísla podmíněnosti matice A	Double (F64)
E	Příznak chyby	Bool
info	Informace o výsledku funkce LAPACKu. Je-li info = -i, pak i-tý argument měl nepřípustnou hodnotu	Long (I32)

ML_DGEES – Výpočet vlastních čísel, Schurovy formy a volitelně matice Schurových vektorů

Symbol bloku

Licence: [MATRIX](#)



Popis funkce

Výstupní reference `yA`, `yWR`, `yWI`, `yVS`, `yWORK` a `yBWORK` jsou vždy nastaveny na odpovídající vstupní reference `uA`, `uWR`, `uWI`, `uVS`, `uWORK` a `uBWORK`. Pokud `HLD = on`, pak není provedeno nic jiného, jinak je interně zavolána funkce LAPACK DGEES:

```
DGEES(sJOBVS, sSORT, SELECT, N, uA, LDA, sdim, uWR, uWI, uVS, LDVS, uWORK,
      LWORK, uBWORK, info);
```

kde parametry funkce DGEES jsou nastaveny následovně:

- Pokud `JOBVS = on` pak je řetězec `sJOBVS` nastaven na "V" jinak je nastaven na "N".
- Pokud `SORT = on` pak je řetězec `sSORT` nastaven na "S" jinak je nastaven na "N".
- `SELECT` je reference na funkci pro třídění vlastních čísel, která v tomto funkčním bloku vrací vždy `on`.
- `N` je počet sloupců matice referencované `uA`.
- `LDA` je hlavní rozměr matice referencované `uA`.
- `sdim` je vrácen počet vlastních čísel, pro které funkce `SELECT` vrací `on`.
- `LDVS` je hlavní rozměr matice referencované `uVS`.
- `LWORK` je počet prvků vektoru referencovaného `uWORK`.
- `info` je návratový kód z funkce DGEES.

Chybový příznak `E` je nastaven na `on`, pokud:

- reference `uA` nebo `uWR` nebo `uWI` nebo `uVS` nebo `uWORK` nebo `uBWORK` není definována (tj. vstup `uA` nebo `uWR` nebo `uWI` nebo `uVS` nebo `uWORK` nebo `uBWORK` není připojen),

- matice referencovaná `uA` není čtvercová,
- počet prvků vektoru referencovaného `uWR`, `uWI` a `uBWORK` je menší než `N`,
- počet sloupců matice referencované `uVS` není roven `N`,
- volání funkce `DGEES` vrátí chybu pomocí funkce `XERBLA`, viz návratový kód `info` a systémový log.

Více informací naleznete v dokumentaci knihovny LAPACK [11] (pouze anglicky).

Tento blok nepropaguje kvalitu signálu. Více informací je uvedeno v sekci 1.4.

Vstup

<code>uA</code>	Vstupní odkaz na matici <code>A</code>	Reference
<code>uWR</code>	Vstupní odkaz na vektor reálných částí vlastních čísel	Reference
<code>uWI</code>	Vstupní odkaz na vektor imaginárních částí vlastních čísel	Reference
<code>uVS</code>	Vstupní reference na ortogonální matici Schurových vektorů	Reference
<code>uWORK</code>	Vstupní odkaz na pracovní vektor <code>WORK</code>	Reference
<code>uBWORK</code>	Vstupní odkaz na Booleovský pracovní vektor <code>WORK</code>	Reference
<code>JOBVS</code>	Je-li <code>true</code> , pak se počítají Schurovy vektory	Bool
<code>SORT</code>	Je-li <code>true</code> , pak se vlastní čísla setřídí	Bool
<code>HLD</code>	Pozastavení	Bool

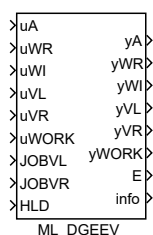
Výstup

<code>yA</code>	Výstupní odkaz na matici <code>A</code>	Reference
<code>yWR</code>	Výstupní odkaz na vektor reálných částí vlastních čísel	Reference
<code>yWI</code>	Výstupní odkaz na vektor imaginárních částí vlastních čísel	Reference
<code>yVS</code>	Výstupní reference na <code>VS</code>	Reference
<code>yWORK</code>	Výstupní odkaz na pracovní vektor <code>WORK</code>	Reference
<code>yBWORK</code>	Výstupní odkaz na Booleovský pracovní vektor <code>WORK</code>	Reference
<code>sdim</code>	Je-li <code>SORT</code> , pak udává počet vlastních čísel pro něž je <code>SELECT</code> <code>true</code> , jinak 0	Long (I32)
<code>E</code>	Příznak chyby	Bool
<code>info</code>	Informace o výsledku funkce LAPACKu. Je-li <code>info = -i</code> , pak <code>i</code> -tý argument měl nepřípustnou hodnotu	Long (I32)

ML_DGEEV – Výpočet vlastních čísel a volitelně levých a/nebo pravých vlastních vektorů

Symbol bloku

Licence: [MATRIX](#)



Popis funkce

Výstupní reference `yA`, `yWR`, `yWI`, `yVL`, `yVR` a `yWORK` jsou vždy nastaveny na odpovídající vstupní reference `uA`, `uWR`, `uWI`, `uVL`, `uVR` a `uWORK`. Pokud `HLD = on` pak není provedeno nic jiného, jinak je interně zavolána funkce LAPACK `DGEEV`:

```
DGEEV(sJOBVL, sJOBVR, N, uA, LDA, uWR, uWI, uVL, LDVL, uVR, LDVR,
      uWORK, LWORK, info);
```

kde parametry funkce `DGEEV` jsou nastaveny následovně:

- Pokud `JOBVL = on`, pak je řetězec `sJOBVL` nastaven na `"V"` jinak je nastaven na `"N"`.
- Pokud `JOBVR = on`, pak je řetězec `sJOBVR` nastaven na `"V"` jinak je nastaven na `"N"`.
- `N` je počet sloupců matice referencované `uA`.
- `LDA`, `LDVL` a `LDVR` jsou hlavní rozměry matic referencovaných `uA`, `uVL` a `uVR`.
- `LWORK` je počet prvků vektoru referencovaného `uWORK`.
- `info` je návratový kód z funkce `DGEEV`.

Chybový příznak `E` je nastaven na `on`, pokud:

- reference `uA` nebo `uWR` nebo `uWI` nebo `uVL` nebo `uVR` nebo `uWORK` není definována (tj. vstup `uA` nebo `uWR` nebo `uWI` nebo `uVL` nebo `uVR` nebo `uWORK` není připojen),
- matice referencovaná `uA` není čtvercová,
- počet prvků vektoru referencovaného `uWR` nebo `uWI` je menší než `N`,

- hlavní rozměry matic referencovaných `uVL` nebo `uVR` nejsou rovny `N`,
- volání funkce `DGEEV` vrátí chybu pomocí funkce `XERBLA`, viz návratový kód `info` a systémový log.

Více informací naleznete v dokumentaci knihovny LAPACK [11] (pouze anglicky).

Tento blok nepropaguje kvalitu signálu. Více informací je uvedeno v sekci 1.4.

Vstup

<code>uA</code>	Vstupní odkaz na matici <code>A</code>	Reference
<code>uWR</code>	Vstupní odkaz na vektor reálných částí vlastních čísel	Reference
<code>uWI</code>	Vstupní odkaz na vektor imaginárních částí vlastních čísel	Reference
<code>uVL</code>	Vstupní reference na matici levých vlastních vektorů	Reference
<code>uVR</code>	Vstupní reference na matici pravých vlastních vektorů	Reference
<code>uWORK</code>	Vstupní odkaz na pracovní vektor <code>WORK</code>	Reference
<code>JOBVL</code>	Je-li <code>true</code> , pak se počítají levé vlastní vektory	Bool
<code>JOBVR</code>	Je-li <code>true</code> , pak se počítají pravé vlastní vektory	Bool
<code>HLD</code>	Pozastavení	Bool

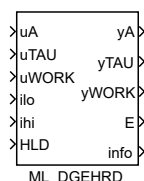
Výstup

<code>yA</code>	Výstupní odkaz na matici <code>A</code>	Reference
<code>yWR</code>	Výstupní odkaz na vektor reálných částí vlastních čísel	Reference
<code>yWI</code>	Výstupní odkaz na vektor imaginárních částí vlastních čísel	Reference
<code>yVL</code>	Výstupní reference na <code>VL</code>	Reference
<code>yVR</code>	Výstupní reference na <code>VR</code>	Reference
<code>yWORK</code>	Výstupní odkaz na pracovní vektor <code>WORK</code>	Reference
<code>E</code>	Příznak chyby	Bool
<code>info</code>	Informace o výsledku funkce LAPACKu. Je-li <code>info = -i</code> , pak <code>i</code> -tý argument měl nepřipustnou hodnotu	Long (I32)

ML_DGEHRD – Redukce reálné obecné matice A na horní Hessenbergovu formu

Symbol bloku

Licence: [MATRIX](#)



Popis funkce

Výstupní reference yA , $yTAU$ a $yWORK$ jsou vždy nastaveny na odpovídající vstupní reference uA , $uTAU$ a $uWORK$. Pokud $HLD = \text{on}$, pak není provedeno nic jiného, jinak je interně zavolána funkce LAPACK DGEHRD:

```
DGEHRD(N, ilo, ihi, uA, LDA, uTAU, uWORK, LWORK, info);
```

kde parametry funkce DGEHRD jsou nastaveny následovně:

- N je počet sloupců čtvercové matice referencované uA .
- Pokud je vstup $ihi \neq 0$, pak je ihi nastaveno na ihi , jinak je ihi nastaveno na $N - 1$.
- LDA je hlavní rozměr matice referencované uA .
- $LWORK$ je počet prvků vektoru referencovaného $uWORK$.
- $info$ je návratový kód z funkce DGEHRD.

Chybový příznak E je nastaven na on , pokud:

- reference uA nebo $uTAU$ nebo $uWORK$ není definována (tj. vstup uA nebo $uTAU$ nebo $uWORK$ není připojen),
- matice referencovaná uA není čtvercová,
- počet prvků vektoru referencovaný pomocí $uTAU$ je menší než $N - 1$,
- volání funkce DGEHRD vrátí chybu pomocí funkce XERBLA, viz návratový kód $info$ a systémový log.

Důrazně upozorňujeme, že indexy ilo a ihi začínají od nuly na rozdíl od FORTRAN verze, kde začínají od jedničky. Více informací naleznete v dokumentaci knihovny LA-

PACK [\[11\]](#) (pouze anglicky).

Tento blok nepropaguje kvalitu signálu. Více informací je uvedeno v sekci [1.4](#).

Vstup

<code>uA</code>	Vstupní odkaz na matici <code>A</code>	Reference
<code>uTAU</code>	Vstupní reference na vektor skalárních faktorů elementárních reflektorů	Reference
<code>uWORK</code>	Vstupní odkaz na pracovní vektor <code>WORK</code>	Reference
<code>ilo</code>	Dolní index (od nuly) řádku a sloupce pracovní submatice	Long (I32)
<code>ihi</code>	Horní index (od nuly) řádku a sloupce pracovní submatice	Long (I32)
<code>HLD</code>	Pozastavení	Bool

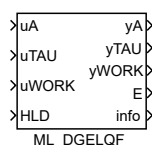
Výstup

<code>yA</code>	Výstupní odkaz na matici <code>A</code>	Reference
<code>yTAU</code>	Výstupní reference na vektor skalárních faktorů elementárních reflektorů	Reference
<code>yWORK</code>	Výstupní odkaz na pracovní vektor <code>WORK</code>	Reference
<code>E</code>	Příznak chyby	Bool
<code>info</code>	Informace o výsledku funkce LAPACKu. Je-li <code>info = -i</code> , pak <code>i</code> -tý argument měl nepřipustnou hodnotu	Long (I32)

ML_DGELQF – Výpočet LQ faktorizace reálné matice A s rozměry $M \times N$

Symbol bloku

Licence: [MATRIX](#)



Popis funkce

Výstupní reference yA , $yTAU$ a $yWORK$ jsou vždy nastaveny na odpovídající vstupní reference uA , $uTAU$ a $uWORK$. Pokud $HLD = on$, pak není provedeno nic jiného, jinak je interně zavolána funkce LAPACK `DGELQF`:

`DGELQF(M, N, uA, LDA, uTAU, uWORK, LWORK, info);`

kde parametry funkce `DGELQF` jsou nastaveny následovně:

- M je počet řádků matice referencované uA .
- N je počet sloupců matice referencované uA .
- LDA je hlavní rozměr matice referencované uA .
- $LWORK$ je počet prvků vektoru referencovaného $uWORK$.
- `info` je návratový kód z funkce `DGELQF`.

Chybový příznak E je nastaven na `on`, pokud:

- reference uA nebo $uTAU$ nebo $uWORK$ není definována (tj. vstup uA nebo $uTAU$ nebo $uWORK$ není připojen),
- počet prvků vektoru referencovaný pomocí $uTAU$ je menší než minimum počtu řádků a počtu sloupců matice referencované uA ,
- volání funkce `DGELQF` vrátí chybu pomocí funkce `XERBLA`, viz návratový kód `info` a systémový log.

Více informací naleznete v dokumentaci knihovny LAPACK [11] (pouze anglicky).

Tento blok nepropaguje kvalitu signálu. Více informací je uvedeno v sekci 1.4.

Vstup

uA	Vstupní odkaz na matici A	Reference
uTAU	Vstupní reference na vektor skalárních faktorů elementárních reflektorů	Reference
uWORK	Vstupní odkaz na pracovní vektor WORK	Reference
HLD	Pozastavení	Bool

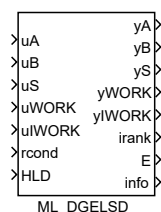
Výstup

yA	Výstupní odkaz na matici A	Reference
yTAU	Výstupní reference na vektor skalárních faktorů elementárních reflektorů	Reference
yWORK	Výstupní odkaz na pracovní vektor WORK	Reference
E	Příznak chyby	Bool
info	Informace o výsledku funkce LAPACKu. Je-li info = -i, pak i-tý argument měl nepřipustnou hodnotu	Long (I32)

ML_DGELSD – Výpočet řešení s minimální normou reálné lineární úlohy nejmenších čtverců

Symbol bloku

Licence: [MATRIX](#)



Popis funkce

Výstupní reference `yA`, `yB`, `yS`, `yWORK` a `yIWORK` jsou vždy nastaveny na odpovídající vstupní reference `uA`, `uB`, `uS`, `uWORK` a `uIWORK`. Pokud `HLD = on` pak není provedeno nic jiného, jinak je interně zavolána funkce LAPACK `DGELSD`:

```
DGELSD(M, N, NRHS, uA, LDA, uB, LDB, uS, rcond, irank, uWORK,
        LWORK, uIWORK, info);
```

kde parametry funkce `DGELSD` jsou nastaveny následovně:

- `M` je počet řádků matice referencované `uA`.
- `N` je počet sloupců matice referencované `uA`.
- `NRHS` je počet sloupců matice referencované `uB`.
- `LDA` a `LDB` jsou hlavní rozměry matic referencovaných `uA` a `uB`.
- `irank` je vrácený efektivní rank matice referencované `uA`.
- `LWORK` je počet prvků vektoru referencovaného `uWORK`.
- `info` je návratový kód z funkce `DGELSD`.

Chybový příznak `E` je nastaven na `on`, pokud:

- reference `uA` nebo `uB` nebo `uS` nebo `uWORK` nebo `uIWORK` není definována (tj. vstup `uA` nebo `uB` nebo `uS` nebo `uWORK` nebo `uIWORK` není připojen),
- počet řádků matice referencované `uB` není roven `M`,
- počet prvků vektoru referencovaný pomocí `uS` je menší než minimum z `M` a `N`,
- počet prvků celočíselného vektoru referencovaného `uIWORK` není dostatečný (viz detaily v dokumentaci LAPACK funkce `DGELSD`),

- volání funkce `DGELSD` vrátí chybu pomocí funkce `XERBLA`, viz návratový kód `info` a systémový log.

Více informací naleznete v dokumentaci knihovny LAPACK [11] (pouze anglicky).

Tento blok nepropaguje kvalitu signálu. Více informací je uvedeno v sekci 1.4.

Vstup

<code>uA</code>	Vstupní odkaz na matici A	Reference
<code>uB</code>	Vstupní odkaz na matici B	Reference
<code>uS</code>	Vstupní reference na vektor singulárních hodnot	Reference
<code>uWORK</code>	Vstupní odkaz na pracovní vektor WORK	Reference
<code>uIWORK</code>	Vstupní odkaz na celočíselný pracovní vektor WORK	Reference
<code>rcond</code>		Double (F64)
<code>HLD</code>	Pozastavení	Bool

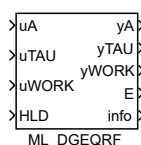
Výstup

<code>yA</code>	Výstupní odkaz na matici A	Reference
<code>yB</code>	Výstupní odkaz na matici B	Reference
<code>yS</code>	Výstupní reference na vektor singulárních hodnot	Reference
<code>yWORK</code>	Výstupní odkaz na pracovní vektor WORK	Reference
<code>yIWORK</code>	Výstupní odkaz na celočíselný pracovní vektor WORK	Reference
<code>irank</code>		Long (I32)
<code>E</code>	Příznak chyby	Bool
<code>info</code>	Informace o výsledku funkce LAPACKu. Je-li <code>info = -i</code> , pak <code>i</code> -tý argument měl nepřipustnou hodnotu	Long (I32)

ML_DGEQRF – Výpočet QR faktorizace reálné matice A s rozměry $M \times N$

Symbol bloku

Licence: [MATRIX](#)



Popis funkce

Výstupní reference yA , $yTAU$ a $yWORK$ jsou vždy nastaveny na odpovídající vstupní reference uA , $uTAU$ a $uWORK$. Pokud $HLD = on$, pak není provedeno nic jiného, jinak je interně zavolána funkce LAPACK DGEQRF:

```
DGEQRF(M, N, uA, LDA, uTAU, uWORK, LWORK, info);
```

kde parametry funkce DGEQRF jsou nastaveny následovně:

- M je počet řádků matice referencované uA .
- N je počet sloupců matice referencované uA .
- LDA je hlavní rozměr matice referencované uA .
- $LWORK$ je počet prvků vektoru referencovaného $uWORK$.
- $info$ je návratový kód z funkce DGEQRF.

Chybový příznak E je nastaven na `on`, pokud:

- reference uA nebo $uTAU$ nebo $uWORK$ není definována (tj. vstup uA nebo $uTAU$ nebo $uWORK$ není připojen),
- počet prvků vektoru referencovaný pomocí $uTAU$ je menší než minimum počtu řádků a počtu sloupců matice referencované uA ,
- volání funkce DGEQRF vrátí chybu pomocí funkce XERBLA, viz návratový kód $info$ a systémový log.

Více informací naleznete v dokumentaci knihovny LAPACK [11] (pouze anglicky).

Tento blok nepropaguje kvalitu signálu. Více informací je uvedeno v sekci 1.4.

Vstup

uA	Vstupní odkaz na matici A	Reference
uTAU	Vstupní reference na vektor skalárních faktorů elementárních reflektorů	Reference
uWORK	Vstupní odkaz na pracovní vektor WORK	Reference
HLD	Pozastavení	Bool

Výstup

yA	Výstupní odkaz na matici A	Reference
yTAU	Výstupní reference na vektor skalárních faktorů elementárních reflektorů	Reference
yWORK	Výstupní odkaz na pracovní vektor WORK	Reference
E	Příznak chyby	Bool
info	Informace o výsledku funkce LAPACKu. Je-li info = -i, pak i-tý argument měl nepřipustnou hodnotu	Long (I32)

ML_DGESDD – Výpočet singulární dekompozice (SVD) reálné matice A s rozměry $M \times N$

Symbol bloku

Licence: [MATRIX](#)

>uA	>yA
>uS	>yS
>uU	>yU
>uVT	>yVT
>uWORK	>yWORK
>uIWORK	>yIWORK
>jobz	>E
>HLD	>info
ML_DGESDD	

Popis funkce

Výstupní reference yA , yS , yU , yVT , $yWORK$ a $yIWORK$ jsou vždy nastaveny na odpovídající vstupní reference uA , uS , uU , uVT , $uWORK$ a $uIWORK$. Pokud $HLD = on$ pak není provedeno nic jiného, jinak je interně zavolána funkce LAPACK DGESDD:

```
DGESDD(sJOBZ, M, N, uA, LDA, uS, uU, LDU, uVT, LDVT, uWORK, LWORK,
       uIWORK, info);
```

kde parametry funkce DGESDD jsou nastaveny následovně:

- Integerový vstup $jobz$ je mapován na řetězec $sJOBZ$: $\{0,1\} \rightarrow "A"$, $\{2\} \rightarrow "S"$, $\{3\} \rightarrow "U"$ a $\{4\} \rightarrow "N"$.
- M je počet řádků matice referencované uA .
- N je počet sloupců matice referencované uA .
- LDA , LDU a $LDVT$ jsou hlavní rozměry matic referencovaných uA , uU a uVT .
- $LWORK$ je počet prvků vektoru referencovaného $uWORK$.
- $info$ je návratový kód z funkce DGESDD.

Chybový příznak E je nastaven na **on**, pokud:

- reference uA nebo uS nebo uU nebo uVT nebo $uWORK$ nebo $uIWORK$ není definována (tj. vstup uA nebo uS nebo uU nebo uVT nebo $uWORK$ nebo $uIWORK$ není připojen),
- počet prvků vektoru referencovaný pomocí uS je menší než $MINMN$, minimum počtu řádků a počtu sloupců matice referencované uA ,
- počet prvků celočíselného vektoru referencovaného pomocí $uIWORK$ je menší než $8 * MINMN$,

- volání funkce `DGESDD` vrátí chybu pomocí funkce `XERBLA`, viz návratový kód `info` a systémový log.

Více informací naleznete v dokumentaci knihovny LAPACK [11] (pouze anglicky).

Tento blok nepropaguje kvalitu signálu. Více informací je uvedeno v sekci 1.4.

Vstup

<code>uA</code>	Vstupní odkaz na matici <code>A</code>	Reference
<code>uS</code>	Vstupní reference na vektor singulárních hodnot	Reference
<code>uU</code>	Vstupní reference na matici obsahující levé singulární vektory matice <code>A</code>	Reference
<code>uVT</code>	Vstupní reference na matici obsahující pravé singulární vektory matice <code>A</code>	Reference
<code>uWORK</code>	Vstupní odkaz na pracovní vektor <code>WORK</code>	Reference
<code>uIWORK</code>	Vstupní odkaz na celočíselný pracovní vektor <code>WORK</code>	Reference
<code>jobz</code>	Specifikuje volby výpočtu	Long (I32)
<code>HLD</code>	Pozastavení	Bool

Výstup

<code>yA</code>	Výstupní odkaz na matici <code>A</code>	Reference
<code>yS</code>	Výstupní reference na vektor singulárních hodnot	Reference
<code>yU</code>	Výstupní reference na matici obsahující levé singulární vektory matice <code>A</code>	Reference
<code>yVT</code>	Výstupní reference na matici obsahující pravé singulární vektory matice <code>A</code>	Reference
<code>yWORK</code>	Výstupní odkaz na pracovní vektor <code>WORK</code>	Reference
<code>yIWORK</code>	Výstupní odkaz na celočíselný pracovní vektor <code>WORK</code>	Reference
<code>E</code>	Příznak chyby	Bool
<code>info</code>	Informace o výsledku funkce LAPACKu. Je-li <code>info = -i</code> , pak <code>i</code> -tý argument měl nepřipustnou hodnotu	Long (I32)

ML_DLACPY – Kopíruje celou nebo část matice do jiné matice

Symbol bloku

Licence: [STANDARD](#)



Popis funkce

Výstupní reference `yA` a `yB` jsou vždy nastaveny na odpovídající vstupní reference `uA` a `uB`. Pokud `HLD = on` pak není provedeno nic jiného, jinak je interně zavolána funkce LAPACK `DLACPY`:

`DLACPY(sUPL0, M, N, uA, LDA, uB, LDA);`

kde parametry funkce `DLACPY` jsou nastaveny následovně:

- Integerový vstup `uplo` je mapován na řetězec `sUPL0`: $\{0, 1\} \rightarrow "A"$, $\{2\} \rightarrow "U"$ a $\{3\} \rightarrow "L"$.
- `M` je počet řádků matice referencované `uA`.
- `N` je počet sloupců matice referencované `uA`.
- `LDA` je hlavní rozměr matice referencované `uA`.

Počet řádků matice referencované `uB` je nastaven na `M` a hlavní rozměr matice referencované `uB` je nastaven na `LDA`

Chybový příznak `E` je nastaven na `on`, pokud:

- reference `uA` nebo `uB` není definována (tj. vstup `uA` nebo `uB` není připojen),
- počet prvků matice referencované `uA` je odlišný od počtu prvků matice referencované `uB`.

Více informací naleznete v dokumentaci knihovny LAPACK [\[11\]](#) (pouze anglicky).

Tento blok nepropaguje kvalitu signálu. Více informací je uvedeno v sekci [1.4](#).

Vstup

<code>uA</code>	Vstupní odkaz na matici A	Reference
<code>uB</code>	Vstupní odkaz na matici B	Reference

uplo	Kopírovaná část matice	Long (I32)
	0 Celá	
	1 Celá	
	2 Horní	
	3 Dolní	
HLD	Pozastavení	Bool

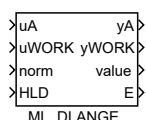
Výstup

yA	Výstupní odkaz na matici A	Reference
yB	Výstupní odkaz na matici B	Reference
E	Příznak chyby	Bool

ML_DLANGE – Výpočet některé z maticových norem obecné matice

Symbol bloku

Licence: [STANDARD](#)



Popis funkce

Výstupní reference `yA` a `yWORK` jsou vždy nastaveny na odpovídající vstupní reference `uA` a `uWORK`. Pokud `HLD = on` pak není provedeno nic jiného, jinak je interně zavolána funkce LAPACK `DLANGE`:

```
value = DLANGE(sNORM, M, N, uA, LDA, uWORK;
```

kde parametry funkce `DLAPCY` jsou nastaveny následovně:

- Celočíslný vstup `norm` je mapován na řetězec `sNORM`: $\{0, 1\} \rightarrow \text{"F"}$ (Frobeniova norma), $\{2\} \rightarrow \text{"M"}$ ($\max(\text{abs}(A(i, j)))$), $\{3\} \rightarrow \text{"1"}$ (jedničková norma) a $\{4\} \rightarrow \text{"I"}$ (nekonečná norma).
- `M` je počet řádků matice referencované `uA`.
- `N` je počet sloupců matice referencované `uA`.
- `LDA` je hlavní rozměr matice referencované `uA`.
- `uWORK` je pracovní vektor o rozměru `LWORK` $\geq M$. `uWORK` je použit pouze pro nekonečnou normu, jinak není referencován.

Chybový příznak `E` je nastaven na `on`, pokud:

- reference `uA` není definována (tj. vstup `uA` není připojen),
- reference `uWORK` není definována pro `norm = 4` (tj. vstup `uWORK` není připojen).

Více informací naleznete v dokumentaci knihovny LAPACK [\[11\]](#) (pouze anglicky).

Použijte blok [MB_DNRM2](#) pro výpočet Frobeniovy normy vektoru.

Tento blok nepropaguje kvalitu signálu. Více informací je uvedeno v sekci [1.4](#).

Vstup

<code>uA</code>	Vstupní odkaz na matici <code>A</code>	Reference
<code>uWORK</code>	Vstupní odkaz na pracovní vektor <code>WORK</code>	Reference
<code>norm</code>	Zvolená maticová norma	$\downarrow 0 \uparrow 4$ Long (I32)
<code>HLD</code>	Pozastavení	Bool

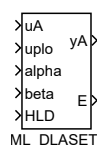
Výstup

yA	Výstupní odkaz na matici A	Reference
yWORK	Výstupní odkaz na pracovní vektor WORK	Reference
value	Návratová hodnota funkce	Double (F64)
E	Příznak chyby	Bool

ML_DLASET – Inicializuje mimodiagonální a diagonální prvky matice na zadané hodnoty

Symbol bloku

Licence: [STANDARD](#)



Popis funkce

Výstupní reference `yA` je vždy nastavena na odpovídající vstupní reference `uA`. Pokud `HLD = on`, pak není provedeno nic jiného, jinak je interně zavolána funkce LAPACK `DLASET`:

```
DLASET(sUPL0, M, N, alpha, beta, uA, LDA);
```

kde parametry funkce `DLASET` jsou nastaveny následovně:

- Integerový vstup `uplo` je mapován na řetězec `sUPL0`: $\{0, 1\} \rightarrow "A"$, $\{2\} \rightarrow "U"$ a $\{3\} \rightarrow "L"$.
- `M` je počet řádků matice referencované `uA`.
- `N` je počet sloupců matice referencované `uA`.
- `LDA` je hlavní rozměr matice referencované `uA`.

Chybový příznak `E` je nastaven na `on`, pokud:

- reference `uA` není definována (tj. vstup `uA` není připojen).

Více informací naleznete v dokumentaci knihovny LAPACK [\[11\]](#) (pouze anglicky).

Tento blok nepropaguje kvalitu signálu. Více informací je uvedeno v sekci [1.4](#).

Vstup

uA	Vstupní odkaz na matici A	Reference
uplo	Kopírovaná část matice	Long (I32)
	0 Celá	
	1 Celá	
	2 Horní	
	3 Dolní	
alpha	Skalární koeficient alpha	Double (F64)
beta	Skalární koeficient beta	Double (F64)
HLD	Pozastavení	Bool

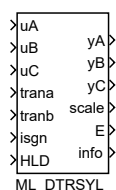
Výstup

yA	Výstupní odkaz na matici A	Reference
E	Příznak chyby	Bool

ML_DTRSYL – Řešení reálné Sylvesterovy rovnice pro kvazitroj- úhelníkové matice **A** a **B**

Symbol bloku

Licence: [MATRIX](#)



Popis funkce

Výstupní reference **yA**, **yB** a **yC** jsou vždy nastaveny na odpovídající vstupní reference **uA**, **uB** a **uC**. Pokud **HLD** = **on**, pak není provedeno nic jiného, jinak je interně zavolána funkce LAPACK **DTRSYL**:

```
DTRSYL(sTRANA, sTRANB, isgn, M, N, uA, LDA, uB, LDB, uC, LDC, uWORK, LWORK,
uIWORK, info);
```

kde parametry funkce **DTRSYL** jsou nastaveny následovně:

- Celočíselný vstup **trana** je mapován na řetězec **sTRANA**: $\{0\} \rightarrow "N"$ a $\{1\} \rightarrow "T"$.
- **M** je počet řádků matice referencované **uA**.
- **N** je počet sloupců matice referencované **uB**.
- **LDA**, **LDB** a **LDC** jsou hlavní rozměry matic referencovaných **uA**, **uB** a **uC**.
- **scale** je vrácený faktor škálování pro zabránění přetečení.
- **info** je návratový kód z funkce **DTRSYL**.

Chybový příznak **E** je nastaven na **on**, pokud:

- reference **uA** nebo **uB** nebo **uC** není definována (tj. vstup **uA** nebo **uB** nebo **uC** není připojen),
- **trana** a **tranb** je menší než 0 nebo větší než 3,
- počet sloupců matice referencované **uA** je odlišný od **M**,
- počet řádků matice referencované **uB** je odlišný od **N**,
- počet řádků matice referencované **uC** je odlišný od **N** nebo počet sloupců této matice je odlišný od **M**,

- volání funkce `DTRSYL` vrátí chybu pomocí funkce `XERBLA`, viz systémový log.

Více informací naleznete v dokumentaci knihovny LAPACK [11] (pouze anglicky).

Tento blok nepropaguje kvalitu signálu. Více informací je uvedeno v sekci 1.4.

Vstup

<code>uA</code>	Vstupní odkaz na matici A		Reference
<code>uB</code>	Vstupní odkaz na matici B		Reference
<code>uC</code>	Vstupní odkaz na matici C		Reference
<code>trana</code>	Transpozice matice A	↓0 ↑3	Long (I32)
<code>tranb</code>	Transpozice matice B	↓0 ↑3	Long (I32)
<code>isgn</code>	Znaménko v rovnici (1 nebo -1)	↓-1 ↑1	Long (I32)
<code>HLD</code>	Pozastavení		Bool

Výstup

<code>yA</code>	Výstupní odkaz na matici A	Reference
<code>yB</code>	Výstupní odkaz na matici B	Reference
<code>yC</code>	Výstupní odkaz na matici C	Reference
<code>scale</code>	Scale	Double (F64)
<code>E</code>	Příznak chyby	Bool
<code>info</code>	Informace o výsledku funkce LAPACKu. Je-li <code>info = -i</code> , pak <code>i</code> -tý argument měl nepřípustnou hodnotu	Long (I32)

MX_AT – Hodnota prvku matice/vektoru

Symbol bloku

Licence: [STANDARD](#)



Popis funkce

Funkční blok **MX_AT** vrací hodnotu (výstup **value**) prvku matice na **i**-tém řádku a **j**-tém sloupci nebo **i**-tém prvku vektoru.

Výstupní reference **yMV** je vždy nastavena na odpovídající vstupní referenci **uMV** k připojené matici/vektoru.

Chybový příznak **E** je nastaven na **on** pokud:

- reference **uMV** není definována (tj. vstup **uMV** není připojen),
- index řádku $i < 0$ nebo $i \geq m$, kde **m** je počet řádků,
- index sloupce $j < 0$ nebo $j \geq n$, kde **n** je počet sloupců. Poznámka: **j** musí být 0 pro vektor.

Tento blok nepropaguje kvalitu signálu. Více informací je uvedeno v sekci [1.4](#).

Vstup

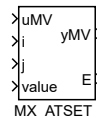
uMV	Vstupní reference na matici nebo vektor		Reference
i	Řádkový index prvku	↓0	Long (I32)
j	Sloupcový index prvku	↓0	Long (I32)

Výstup

yMV	Výstupní reference na matici nebo vektor		Reference
value	Hodnota prvku v pozici (i,j)		Long (I32)
E	Příznak chyby		Bool

MX_ATSET – Nastavení hodnoty prvku matice/vektoru

Symbol bloku

Licence: [STANDARD](#)

Popis funkce

Funční blok **MX_ATSET** nastavuje hodnotu (vstup **value**) prvku matice na **i**-tém řádku a **j**-tém sloupci nebo na **i**-tém prvku vektoru.

Výstupní reference **yMV** je vždy nastavena na odpovídající vstupní referenci **uMV** k připojené matici/vektoru.

Chybový příznak **E** je nastaven na **on**, pokud:

- reference **uMV** není definována (tj. vstup **uMV** není připojen),
- index řádku $i < 0$ nebo $i \geq m$, kde **m** je počet řádků,
- index sloupce $j < 0$ nebo $j \geq n$, kde **n** je počet sloupců. Poznámka: **j** musí být 0 pro vektor.

Tento blok nepropaguje kvalitu signálu. Více informací je uvedeno v sekci [1.4](#).

Vstup

			Reference
uMV	Vstupní reference na matici nebo vektor		
i	Řádkový index prvku	↓0	Long (I32)
j	Sloupcový index prvku	↓0	Long (I32)
value	Hodnota, která má být nastavena do prvku v pozici (i,j)		Long (I32)

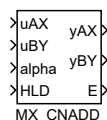
Výstup

		Reference
yMV	Výstupní reference na matici nebo vektor	
E	Příznak chyby	Bool

MX_CNADD – Přičte skalár ke každému prvku matice/vektoru

Symbol bloku

Licence: [STANDARD](#)



Popis funkce

Funkční blok MX_CNADD přičítá hodnotu vstupu **alpha** ke každému prvku matice/vektoru referencovaného **uAX** a výsledek je uložen do matice/vektoru referencovaného **uBY**.

Výstupní reference **yAX** a **yBY** jsou vždy nastaveny na odpovídající vstupní reference **uAX** a **uBY**. Rozměry matice/vektoru referencovaného **uBY** jsou nastaveny na rozměry matice/vektoru referencovaného **uAX**, pokud jsou odlišné.

Nastavením **HLD = on** se výpočet zastaví, ale reference se předávají dál. Chybový příznak **E** je nastaven na **on**, pokud:

- reference **uAX** nebo **uBY** není definována (tj. vstup **uAX** nebo **uBY** není připojen),
- počet alokovaných prvků matice/vektoru referencovaného **uAX** je odlišný od počtu alokovaných prvků matice/vektoru referencovaného **uBY**.

Tento blok nepropaguje kvalitu signálu. Více informací je uvedeno v sekci [1.4](#).

Vstup

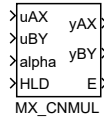
uAX	Vstupní reference na matici A nebo vektor X	Reference
uBY	Vstupní reference na matici B nebo vektor Y	Reference
alpha	Skalární koeficient alpha	Double (F64)
HLD	Pozastavení	Bool

Výstup

yAX	Výstupní reference na matici A nebo vektor X	Reference
yBY	Výstupní reference na matici B nebo vektor Y	Reference
E	Příznak chyby	Bool

MX_CNMUL – Vynásobí matici/vektor skalárem

Symbol bloku

Licence: [STANDARD](#)

Popis funkce

Funkční blok MX_CNMUL násobí hodnotu vstupu **alpha** s každým prvkem matice/vektoru referencovaného **uAX** a výsledek je uložen do matice/vektoru referencovaného **uBY**.

Výstupní reference **yAX** a **yBY** jsou vždy nastaveny na odpovídající vstupní reference **uAX** a **uBY**. Rozměry matice/vektoru referencovaného **uBY** jsou nastaveny na rozměry matice/vektoru referencovaného **uAX**, pokud jsou odlišné.

Nastavením **HLD = on** se výpočet zastaví, ale reference se předávají dál. Chybový příznak **E** je nastaven na **on**, pokud:

- reference **uAX** nebo **uBY** není definována (tj. vstup **uAX** nebo **uBY** není připojen),
- počet alokovaných prvků matice/vektoru referencovaného **uAX** je odlišný od počtu alokovaných prvků matice/vektoru referencovaného **uBY**.

Tento blok nepropaguje kvalitu signálu. Více informací je uvedeno v sekci [1.4](#).

Vstup

uAX	Vstupní reference na matici A nebo vektor X	Reference
uBY	Vstupní reference na matici B nebo vektor Y	Reference
alpha	Skalární koeficient alpha	Double (F64)
HLD	Pozastavení	Bool

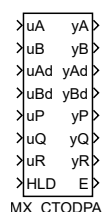
Výstup

yAX	Výstupní reference na matici A nebo vektor X	Reference
yBY	Výstupní reference na matici B nebo vektor Y	Reference
E	Příznak chyby	Bool

MX_CTODPA – Diskretizace spojitého modelu (A,B) do (Ad,Bd) s využitím Padéových aproximací

Symbol bloku

Licence: [STANDARD](#)



Popis funkce

Funkční blok MX_CTODPA diskretizuje model spojitého stavového prostoru pomocí Padéových aproximací matic exponenciálů a jejich integrálů a techniky škálování ([8]). Použitá technika je podobná metodě 3 Scaling and squaring popsané v [12].

Výstupní reference yA, yB, yAd, yBd, yP, yQ a yR jsou vždy nastaveny na odpovídající vstupní reference uA, uB, uAd, uBd, uP, uQ a uR. Pokud HLD = on, pak není provedeno nic jiného, jinak je volána interně funkce mCtoD:

```
mCtoD(nRes, uAd, uBd, uA, uB, N, M, is, Ts, eps, uP, uQ, uR);
```

kde parametry mCtoD jsou nastaveny následovně:

- nRes je návratový kód z funkce mCtoD.
- N je počet řádků čtvercové matice systému referencované uA.
- M je počet sloupců vstupní matice referencované uB.
- Ts je vzorkovací perioda pro diskretizaci, která je rovna vzorkovací periodě úlohy obsahující tento funkční blok.

Chybový příznak E je nastaven na on, pokud:

- reference uA nebo uB nebo uAd nebo uBd nebo uP nebo uQ nebo uR není definována (tj. vstup uA nebo uB nebo uAd nebo uBd nebo uP nebo uQ nebo uR není připojen),
- počet sloupců matice referencované uA není roven N,
- počet řádků matice referencované uB není roven M,
- počet prvků jakékoli matice referencované uAd, uP, uQ nebo uR je menší než $N * N$,
- počet prvků matice referencované uBd je menší než $N * M$,

- návratový kód `nRes` funkce `mCtoD` není roven nule.

Tento blok nepropaguje kvalitu signálu. Více informací je uvedeno v sekci 1.4.

Vstup

<code>uA</code>	Vstupní odkaz na matici A	Reference
<code>uB</code>	Vstupní odkaz na matici B	Reference
<code>uAd</code>	Vstupní reference na diskretizovanou matici A	Reference
<code>uBd</code>	Vstupní reference na diskretizovanou matici B	Reference
<code>uP</code>	Vstupní reference na pomocnou matici	Reference
<code>uQ</code>	Vstupní reference na pomocnou matici	Reference
<code>uR</code>	Vstupní reference na pomocnou matici	Reference
<code>HLD</code>	Pozastavení	Bool

Parametr

<code>is</code>	Řád Padéovy aproximace	↓0 ↑4 ⊙2	Long (I32)
<code>eps</code>	Přesnost aproximace	↓1e-20 ↑0.001 ⊙1e-15	Double (F64)

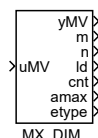
Výstup

<code>yA</code>	Výstupní odkaz na matici A	Reference
<code>yB</code>	Výstupní odkaz na matici B	Reference
<code>yAd</code>	Výstupní reference na diskretizovanou matici A	Reference
<code>yBd</code>	Výstupní reference na diskretizovanou matici B	Reference
<code>yP</code>	Výstupní reference na pomocnou matici	Reference
<code>yQ</code>	Výstupní reference na pomocnou matici	Reference
<code>yR</code>	Výstupní reference na pomocnou matici	Reference
<code>E</code>	Příznak chyby	Bool

MX_DIM – Dimenze matice/vektoru

Symbol bloku

Licence: [STANDARD](#)



Popis funkce

Funkční blok **MX_DIM** nastaví své výstupy na rozměry matice nebo vektoru referencovaného **uMV**.

Výstupní reference **yMV** je vždy nastavena na odpovídající vstupní referenci **uMV**. Chybový příznak **E** je nastaven na **on**, pokud reference **uMV** není definována (tj. vstup **uMV** není připojen).

Tento blok nepropaguje kvalitu signálu. Více informací je uvedeno v sekci [1.4](#).

Vstup

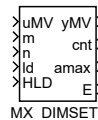
uMV	Vstupní reference na matici nebo vektor	Reference

Výstup

yMV	Výstupní reference na matici nebo vektor	Reference
m	Počet řádků matice	Long (I32)
n	Počet sloupců matice	Long (I32)
ld	Vedoucí dimenze ($\geq$ počtu řádků)	Long (I32)
cnt	Počet použitých prvků matice	Long (I32)
amax	Počet alokovaných prvků ($\geq$ počtu řádků * počet sloupců)	Long (I32)
etype	Typ prvků	Long (I32)
	1 Bool	
	2 Byte (U8)	
	3 Short (I16)	
	4 Long (I32)	
	5 Word (U16)	
	6 DWord (U32)	
	7 Float (F32)	
	8 Double (F64)	
	10 Large (I64)	

MX_DIMSET – Nastavení dimenze matice/vektoru

Symbol bloku

Licence: [STANDARD](#)

Popis funkce

Funkční blok **MX_DIMSET** nastaví počet řádků vektoru na **m** nebo počet řádků **m**, počet sloupců **n** a vedoucí rozměr **ld** matice referencované pomocí **uMV**. Pokud některý z vstupů **m**, **n**, **ld** není připojen, jeho původní hodnota je zachována.

Výstup **cnt** obsahuje skutečný počet obsazených prvků matice/vektoru a je určen vzorcem

$$\text{cnt} = \text{ld} * (\text{n} - 1) + \text{m} \leq \text{amax} ,$$

kde výstup **amax** je alokovaný počet prvků matice/vektoru. Pokud je tato nerovnice splněna, je výstup **cnt** nastaven do struktury matice/vektoru a může být získán pomocí bloku **MX_DIM**, jinak hodnota **cnt** ukazuje minimální nutný počet prvků matice/vektoru.

Výstupní reference **yMV** je vždy nastavena na odpovídající vstupní referenci **uMV**. Chybový příznak **E** je nastaven na **on**, pokud:

- reference **uMV** není definována (tj. vstup **uMV** není připojen),
- počet řádků $\text{m} < 1$ nebo $\text{m} > \text{ld}$,
- počet sloupců $\text{n} < 1$,
- požadovaný počet prvků $\text{cnt} > \text{amax}$.

Tento blok nepropaguje kvalitu signálu. Více informací je uvedeno v sekci [1.4](#).

Vstup

uMV	Vstupní reference na matici nebo vektor	Reference
m	Počet řádků matice	Long (I32)
n	Počet sloupců matice	Long (I32)
ld	Vedoucí dimenze ($\geq$ počtu řádků)	Long (I32)

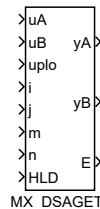
Výstup

yMV	Výstupní reference na matici nebo vektor	Reference
cnt	Počet použitých prvků matice	Long (I32)

amax	Počet alokovaných prvků ($\geq$ počtu řádků * počet sloupců)	Long (I32)
E	Příznak chyby	Bool

MX_DSAGET – Uložení submatice A do matice B

Symbol bloku

Licence: [STANDARD](#)

Popis funkce

Obecně platí, že funkční blok **MX_DSAGET** zkopíruje podmatici matice referencované pomocí **uA** do matice referencované pomocí **uB**.

Výstupní reference **yA** a **yB** jsou vždy nastaveny na odpovídající vstupní reference **uA** a **uB**. Pokud je **HLD = on**, pak se nic nekopíruje, jinak se zkopíruje podmatice matice referencované pomocí **uA** začínající na řádku s indexem 0 a sloupci s indexem 0 obsahující **M** řádků a **N** sloupců (s ohledem na hodnotu vstupu **uplo**) do matice referencované pomocí **uB**. Uvedené proměnné mají následující význam:

- Pokud je vstup $i \leq 0$, pak je **I** nastaveno na 0, jinak pokud je $i \geq MA$, pak je **I** nastaveno na $MA - 1$, jinak je **I** nastaveno na **i**, kde **MA** je počet řádků matice referencované pomocí **uA**.
- Pokud je vstup $j \leq 0$, pak je **J** nastaveno na 0, jinak pokud je $j \geq NA$, pak je **J** nastaveno na $NA - 1$, jinak je **J** nastaveno na **j**, kde **NA** je počet sloupců matice referencované pomocí **uA**.
- Počet zkopírovaných řádků **M** je nastaven ve dvou krocích. Nejprve je **M** nastaveno na minimum z $MA - I$ a počtu řádků matice referencované pomocí **uB**. Dále, pokud $m > 0$, je **M** nastaveno na minimum z **m** a **M**.
- Počet zkopírovaných sloupců **N** je nastaven ve dvou krocích. Nejprve je **N** nastaveno na minimum z $NA - J$ a počtu sloupců matice referencované pomocí **uB**. Dále, pokud $n > 0$, je **N** nastaveno na minimum z **n** a **N**.

Chybový příznak **E** je nastaven na **on**, pokud:

- reference **uA** nebo **uB** není definována (tj. vstup **uA** nebo **uB** není připojen),
- **uplo** je menší než 0 nebo větší než 3,
- počet prvků matice referencované pomocí **uB** je menší než $M * N$.

Tento blok nepropaguje kvalitu signálu. Více informací je uvedeno v sekci [1.4](#).

Vstup

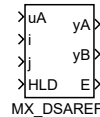
uA	Vstupní odkaz na matici A	Reference
uB	Vstupní odkaz na matici B	Reference
uplo	Kopírovaná část matice	Long (I32)
	0 Celá	
	1 Celá	
	2 Horní	
	3 Dolní	
i	Index prvního řádku podmatice	Long (I32)
j	Index prvního sloupce podmatice	Long (I32)
m	Počet řádků matice	Long (I32)
n	Počet sloupců matice	Long (I32)
HLD	Pozastavení	Bool

Výstup

yA	Výstupní odkaz na matici A	Reference
yB	Výstupní odkaz na matici B	Reference
E	Příznak chyby	Bool

MX_DSAREF – Nastavení odkazu na submatici A do matice B

Symbol bloku

Licence: [STANDARD](#)**Popis funkce**

Funkční blok **MX_DSAREF** vytvoří referenci **yB** na podmatici matice referencované pomocí **uA**. Tato operace je velmi rychlá, protože se žádný prvek matice nekopíruje.

Výstupní reference **yA** je vždy nastavena na odpovídající vstupní referenci **uA**, výstupní reference **yB** je vytvořena uvnitř každé instance tohoto funkčního bloku. Pokud je **HLD = on**, pak se žádná jiná operace neprovádí, jinak se vytvoří reference na matici **yB** s následujícími vlastnostmi:

- Počet řádků podmatice je nastaven na $M-i$, kde M je počet řádků matice referencované pomocí **uA**.
- Počet sloupců podmatice je nastaven na $N-j$, kde N je počet sloupců matice referencované pomocí **uA**.
- První prvek na pozici $(0,0)$ podmatice je prvek matice referencované pomocí **uA** na pozici (i, j) , všechny indexy jsou indexovány od nuly.
- Matice referencovaná pomocí **yB** má stejný vedoucí rozměr jako matice referencovaná pomocí **uA**.

Chybový příznak **E** je nastaven na **on**, pokud:

- reference **uA** není definována (tj. vstup **uA** není připojen),
- $0 > i \geq M$.
- $0 > j \geq N$.

Tento blok nepropaguje kvalitu signálu. Více informací je uvedeno v sekci [1.4](#).

Vstup

		Reference
uA	Vstupní odkaz na matici A	
i	Index prvního řádku podmatice	Long (I32)
j	Index prvního sloupce podmatice	Long (I32)
HLD	Pozastavení	Bool

Parametr

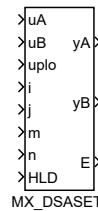
ay	Výstupní reference podmatice	⊙[0 0]	Double (F64)

Výstup

yA	Výstupní odkaz na matici A	Reference
yB	Výstupní odkaz na matici B	Reference
E	Příznak chyby	Bool

MX_DSASET – Uložení matice A do submatice v B

Symbol bloku

Licence: [STANDARD](#)

Popis funkce

Obecně platí, že funkční blok **MX_DSASET** zkopíruje matici referencovanou pomocí **uA** do podmaticy matice referencované pomocí **uB**.

Výstupní reference **yA** a **yB** jsou vždy nastaveny na odpovídající vstupní reference **uA** a **uB**. Pokud je **HLD = on**, pak se nic nekopíruje, jinak se zkopíruje podmatice matice referencované pomocí **uA** začínající řádkem s indexem od nuly **I** a sloupcem s indexem od nuly **J** obsahující **M** řádků a **N** sloupců do matice referencované pomocí **uB**. Uvedené proměnné mají následující významy:

- Pokud je vstup $i \leq 0$ pak je **I** nastaveno na 0, jinak pokud je $i \geq MB$ pak je **I** nastaveno na **MB - 1** jinak je **I** nastaveno na **i**, kde **MB** je počet řádků matice referencované pomocí **uB**.
- Pokud je vstup $j \leq 0$ pak je **J** nastaveno na 0, jinak pokud je $j \geq NB$ pak je **J** nastaveno na **NB - 1** jinak je **J** nastaveno na **j**, kde **NB** je počet sloupců matice referencované pomocí **uB**.
- Počet zkopírovaných řádků **M** je nastaven ve dvou krocích. Nejprve je **M** nastaveno na minimum z **MA - I** a počtu řádků matice referencované pomocí **uB**. Druhý, pokud **m > 0** pak je **M** nastaveno na minimum z **m** a **M**.
- Počet zkopírovaných sloupců **N** je nastaven ve dvou krocích. Nejprve je **N** nastaveno na minimum z **NA - J** a počtu sloupců matice referencované pomocí **uB**. Druhý, pokud **n > 0** pak je **N** nastaveno na minimum z **n** a **N**.

Chybový příznak **E** je nastaven na **on**, pokud:

- reference **uA** nebo **uB** není definována (tj. vstup **uA** nebo **uB** není připojen),
- **uplo** je menší než 0 nebo větší než 3,
- počet prvků matice referencované pomocí **uB** je menší než **M * N**.

Tento blok nepropaguje kvalitu signálu. Více informací je uvedeno v sekci [1.4](#).

Vstup

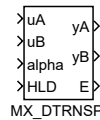
uA	Vstupní odkaz na matici A	Reference
uB	Vstupní odkaz na matici B	Reference
uplo	Kopírovaná část matice	Long (I32)
	0 Celá	
	1 Celá	
	2 Horní	
	3 Dolní	
i	Index prvního řádku podmatice	Long (I32)
j	Index prvního sloupce podmatice	Long (I32)
m	Počet řádků matice	Long (I32)
n	Počet sloupců matice	Long (I32)
HLD	Pozastavení	Bool

Výstup

yA	Výstupní odkaz na matici A	Reference
yB	Výstupní odkaz na matici B	Reference
E	Příznak chyby	Bool

MX_DTRNSP – Transpozice obecné matice: $B := \alpha * A^T$

Symbol bloku

Licence: [STANDARD](#)**Popis funkce**

Funkční blok `MX_DTRNSP` uloží skalární násobek obecné (tj. obdélníkové) matice referencované pomocí `uA` do matice referencované pomocí `uB`.

Výstupní reference `yA` a `yB` jsou vždy nastaveny na odpovídající vstupní reference `uA` a `uB`. Pokud je `HLD = on`, pak se nic jiného neděje, jinak je interně zavolána funkce `X_DTRNSP` podobná funkci BLAS:

```
X_DTRNSP(M, N, ALPHA, uA, LDA, uB, LDB);
```

kde parametry funkce `X_DTRNSP` jsou nastaveny následovně:

- `M` je počet řádků matice referencované pomocí `uA`.
- `N` je počet sloupců matice referencované pomocí `uA`.
- Pokud je vstup `alpha` rovno 0, pak je `ALPHA` nastaveno na 1, jinak je `ALPHA` nastaveno na `alpha`.
- `LDA` a `LDB` jsou vedoucí rozměry matic referencovaných pomocí `uA` a `uB`.

Chybový příznak `E` je nastaven na `on`, pokud:

- reference `uA` nebo `uB` není definována (tj. vstup `uA` nebo `uB` není připojen),
- volání funkce `X_DTRNSP` vrátí chybu pomocí funkce `XERBLA`, viz system log.

Tento blok nepropaguje kvalitu signálu. Více informací je uvedeno v sekci [1.4](#).

Vstup

<code>uA</code>	Vstupní odkaz na matici A	Reference
<code>uB</code>	Vstupní odkaz na matici B	Reference
<code>alpha</code>	Skalární koeficient alpha	Double (F64)
<code>HLD</code>	Pozastavení	Bool

Výstup

yA	Výstupní odkaz na matici A
yB	Výstupní odkaz na matici B
E	Příznak chyby

Reference
Reference
Bool

MX_DTRNSQ – Transpozice čtvercové matice na místě: $A := \alpha * A^T$

Symbol bloku

Licence: [STANDARD](#)



Popis funkce

Funkční blok **MX_DTRNSQ** transponuje skalární násobek čtvercové matice referencované pomocí **uA** na místě.

Výstupní reference **yA** je vždy nastavena na odpovídající vstupní referenci **uA**. Pokud je **HLD = on** pak se nic jiného neděje, jinak je interně zavolána funkce **X_DTRNSQ** podobná funkci **BLAS**:

```
X_DTRNSQ(N, ALPHA, uA, LDA);
```

kde parametry funkce **X_DTRNSQ** jsou nastaveny následovně:

- **N** je počet řádků a sloupců matice referencované pomocí **uA**.
- Pokud je vstup **alpha** rovno 0, pak je **ALPHA** nastaveno na 1, jinak je **ALPHA** nastaveno na **alpha**.
- **LDA** je vedoucí rozměr matice referencované pomocí **uA**.

Chybový příznak **E** je nastaven na **on**, pokud:

- reference **uA** není definována (tj. vstup **uA** není připojen),
- matice referencovaná pomocí **uA** není čtvercová,
- volání funkce **X_DTRNSQ** vrátí chybu pomocí funkce **XERBLA**, viz systémový log.

Tento blok nepropaguje kvalitu signálu. Více informací je uvedeno v sekci [1.4](#).

Vstup

uA	Vstupní odkaz na matici A	Reference
alpha	Skalární koeficient alpha	Double (F64)
HLD	Pozastavení	Bool

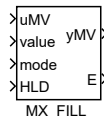
Výstup

yA Výstupní odkaz na matici A
E Příznak chyby

Reference
Bool

MX_FILL – Vyplnění reálné matice/vektoru

Symbol bloku

Licence: [STANDARD](#)

Popis funkce

Funkční blok **MX_FILL** vyplní prvky matice nebo vektoru referencovaného pomocí **uMV** podle vstupu **mode**.

Výstupní reference **yMV** je vždy nastavena na odpovídající vstupní reference **uMV**. Nastavením **HLD = on** se funkce bloku zastaví, ale reference se předávají dál. Chybový příznak **E** je nastaven na **on**, pokud:

- reference **uMV** není definována (tj. vstup **uMV** není připojen),
- $0 > \text{mode} > 4$.

Tento blok nepropaguje kvalitu signálu. Více informací je uvedeno v sekci [1.4](#).

Vstup

uMV	Vstupní reference na matici nebo vektor	Reference
value	Hodnota, kterou bude plněna matice/vektor	Double (F64)
mode	Způsob vyplnění	Long (I32)
	0 Hodnota	
	1 Hodnota	
	2 Jedničky	
	3 Hodnota na diagonálu	
	4 Jednotková matice	
HLD	Pozastavení	Bool

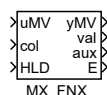
Výstup

yMV	Výstupní reference na matici nebo vektor	Reference
E	Příznak chyby	Bool

MX_FNX – Maticové a vektorové skalární funkce

Symbol bloku

Licence: [STANDARD](#)



Popis funkce

Blok MX_FNX na výstup `val` vrátí výsledek funkce zadané parametrem `efunc`. Funkce je aplikována na prvky matice nebo vektoru referencovaného pomocí `uMV`. Pro matice lze vybrat konkrétní sloupec vstupem `col`. Nastavením `col = -1` se bere v úvahu celá matice. V závislosti na parametru `efunc` je na výstupu `aux` zobrazována pomocná hodnota:

- `efunc = 1:Minimum`: `aux` obsahuje index prvku s minimální hodnotou,
- `efunc = 2:Maximum`: `aux` obsahuje index prvku s maximální hodnotou,
- `efunc = 3:Sum`: `aux` obsahuje počet sčítaných prvků,
- `efunc = 4:Average`: `aux` obsahuje počet průměrovaných prvků,
- `efunc = 5:Standard deviation`: `aux` obsahuje počet prvků pro výpočet směrodatné odchylky,
- `efunc = 6:RMS`: `aux` obsahuje počet prvků pro výpočet středního kvadrátu.

Výstupní reference `yMV` je vždy nastavena na odpovídající vstupní reference `uMV`. Nastavením `HLD = on` se výpočet zastaví, ale reference se předávají dál. Chybový příznak `E` je nastaven na `on`, pokud:

- reference `uMV` není definována (tj. vstup `uMV` není připojen),
- vstup `col` je mimo rozsah sloupců matice referencované `uMV`.

Tento blok nepropaguje kvalitu signálu. Více informací je uvedeno v sekci [1.4](#).

Vstup

uMV	Vstupní reference na matici nebo vektor	Reference
col	Číslo sloupce (-1 pro celou matici)	Long (I32)
HLD	Pozastavení	Bool

Parametr

<code>efunc</code>	Typ funkce	⊙1 Long (I32)
	1 Minimum	
	2 Maximum	
	3 Součet	
	4 Průměr	
	5 Směrodatná odchylka	
	6 Střední kvadrát	
<code>NZ</code>	Zpracovat jen nenulové hodnoty	Bool
<code>tol</code>	Tolerance pro detekci nuly	Double (F64)

Výstup

<code>yMV</code>	Výstupní reference na matici nebo vektor	Reference
<code>val</code>	Hodnota funkce	Double (F64)
<code>aux</code>	Vedlejší hodnota funkce	Long (I32)
<code>E</code>	Příznak chyby	Bool

MX_MAT – Blok pro uložení dat matice

Symbol bloku

Licence: [STANDARD](#)



Popis funkce

Funkční blok MX_MAT alokuje paměť (během inicializace bloku) pro $m * n$ prvků typu určeného parametrem **etype** matice referencované pomocí výstupu **yMat**. Vedoucí rozměr matice může být nastaven parametrem **ld**. Pokud $ld < m$, pak je vedoucí rozměr nastaven na m .

Tento blok nepropaguje kvalitu signálu. Více informací je uvedeno v sekci [1.4](#).

Parametr

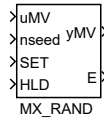
m	Počet řádků matice	↓1 ↑10000000000 ⊙10	Long (I32)
n	Počet sloupců matice	↓1 ↑10000000000 ⊙10	Long (I32)
ld	Vedoucí dimenze ($\geq$ počtu řádků)	↓0 ↑10000000000	Long (I32)
etype	Typ prvků	⊙8	Long (I32)
	1 Bool		
	2 Byte (U8)		
	3 Short (I16)		
	4 Long (I32)		
	5 Word (U16)		
	6 DWord (U32)		
	7 Float (F32)		
	8 Double (F64)		
	10 Large (I64)		

Výstup

yMat	Výstupní reference na matici	Reference

MX_RAND – Náhodně vygenerovaná matice nebo vektor

Symbol bloku

Licence: [STANDARD](#)

Popis funkce

Funkční blok **MX_RAND** vyplní prvky matice nebo vektoru referencovaného pomocí **uMV** náhodnými hodnotami.

Výstupní reference **yMV** je vždy nastavena na odpovídající vstupní reference **uMV**. Pokud je **HLD = on**, pak se nic jiného neděje, jinak se vygeneruje matice nebo vektor pseudo-náhodných hodnot referencovaných vstupem **uMV**. Hodnoty jsou generovány podle následujících pravidel:

- Pokud parametr **BIP** je **on**, pak jsou vygenerované prvky v intervalu $[-\text{scale}; \text{scale}]$, jinak jsou v intervalu $[0; \text{scale}]$.
- Prvky jsou generovány interně pomocí standardní C funkce **rand()**, která generuje pseudo-náhodná čísla v rozsahu od 0 do **RAND_MAX**. Poznámka: hodnota **RAND_MAX** může být závislá na platformě (a měla by být alespoň 32767).
- Vzestupná hrana na výstupu **SET** způsobí, že se před generováním náhodných prvků zavolá standardní C funkce **srand(nseed)** (inicializuje generátor pseudo-náhodných čísel hodnotou **nseed**). Stejně sekvence pseudo-náhodných čísel jsou generovány po volání **srand(nseed)** pro stejné hodnoty **nseed**.

Chybový příznak **E** je nastaven na **on**, pokud reference **uMV** není definována (tj. vstup **uMV** není připojen).

Tento blok nepropaguje kvalitu signálu. Více informací je uvedeno v sekci [1.4](#).

Vstup

		Reference
uMV	Vstupní reference na matici nebo vektor	
nseed	Násada generátoru náhodných čísel	Long (I32)
SET	Nastav na náběžnou hranu počáteční hodnotu generátoru náhodných čísel na nseed	Bool
HLD	Pozastavení	Bool

Parametr

<code>BIP</code>	Příznak náhodných hodnot s oběma polaritami	<code>Bool</code>
<code>scale</code>	Násobitel náhodných hodnot	⊙1.0 <code>Double (F64)</code>

Výstup

<code>yMV</code>	Výstupní reference na matici nebo vektor	<code>Reference</code>
<code>E</code>	Příznak chyby	<code>Bool</code>

MX_REFCOPY – Kopírování vstupních odkazů na matice A a B do jejich výstupních odkazů

Symbol bloku

Licence: [STANDARD](#)



Popis funkce

Funkční blok **MX_REFCOPY** je administrativní blok balíčku **MATRIX**. Nedělá nic jiného, než že zkopíruje vstupní reference **uA** a **uB** do odpovídajících výstupních referencí **yA** a **yB**.

Vhodné zařazení tohoto bloku do schématu funkčního bloku může podstatně ovlivnit pořadí vykonávání bloků, což může být velmi výhodné zejména v kombinaci s bloky jako např. **MX_DSAREF**.

Tento blok nepropaguje kvalitu signálu. Více informací je uvedeno v sekci [1.4](#).

Vstup

uA	Vstupní odkaz na matici A	Reference
uB	Vstupní odkaz na matici B	Reference

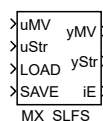
Výstup

yA	Výstupní odkaz na matici A	Reference
yB	Výstupní odkaz na matici B	Reference

MX_SLFS – Ukládání a čtení matice/vektoru do souboru nebo textového řetězce

Symbol bloku

Licence: [STANDARD](#)



Popis funkce

Blok umožňuje konvertovat matici nebo vektor do textové podoby a naopak. Matice se přivádí jako odkaz na vstup **uMV**. Výstup **yMV** odkazuje na stejnou matici, jako vstup **uMV** a je určen pro řetězení maticových bloků ve správném pořadí, jak je obvyklé u všech bloků ze skupiny **MATRIX**.

Text může být buď na vstupu **uStr** (resp. výstupu **yStr** pro opačný směr konverze) nebo v souboru. Pokud je text v souboru, tak jeho název je text připojený na vstup **uStr**. Pro název souboru platí obvyklá pravidla systému **REXYGEN**, tj. je relativně k datadir a není dovoleno **../** pro opuštění adresáře. Pokud je vstup **uStr** nepřipojený (nebo prázdný text), použije se pro jméno souboru název bloku s celou cestou (tj. včetně jména tasku a všech subsystémů) s příponou **.dat**.

Formát matice v textovém souboru nebo v textovém vstupu a výstupu určuje parametr **format**. Podpořeno je anglické i české CSV (tj. sloupce odděleny čárkou nebo středníkem), formát JSON (vytvořený firmou Google a často používaný ve webových aplikacích) a formát používaný MATLABem (pro zadávání matice ve skriptech MATLABu).

Konverze z textové podoby do matice/vektoru nebo naopak může probíhat v každém kroku algoritmu nebo je spouštěna pomocí vstupů **LOAD** a **SAVE**. Přesný způsob je určen parametrem **mode**:

- **1: aktivované úrovni, do/z souboru:** data jsou konvertována ze souboru do matice při **LOAD = on** a z matice do souboru při **SAVE = on**; pokud jsou aktivní oba signály, je to chyba a žádná akce se neprovede,
- **2: aktivované hranou, do/z souboru :** data jsou konvertována ze souboru do matice při náběžné hraně (**off→on**) na vstupu **LOAD** a z matice do souboru při náběžné hraně **SAVE**; pokud jsou náběžné hrany na obou signálech, je to chyba a žádná akce se neprovede,
- **3: aktivované úrovni, do/z textu:** data jsou konvertována ze vstupu **uStr** do matice při **LOAD = on** a z matice na výstup **yStr** při **SAVE = on**; pokud jsou aktivní oba signály, je to chyba a žádná akce se neprovede,

- **4: aktivované hranou, do/z textu:** data jsou konvertována ze vstupu `uStr` do matice při náběžné hraně na vstupu `LOAD` a z matice na výstup `yStr` při náběžné hraně `SAVE`; pokud jsou náběžné hrany na obou signálech, je to chyba a žádná akce se neprovede,
- **5: trvale text do matice:** data jsou konvertována ze vstupu `uStr` do matice v každém kroku algoritmu,
- **6: trvale matice do textu:** data jsou konvertována z matice na výstup `yStr` v každém kroku algoritmu,
- **7: trvale soubor do matice:** data jsou konvertována ze souboru do matice v každém kroku algoritmu,
- **8: trvale matice do souboru:** data jsou konvertována z matice do souboru v každém kroku algoritmu.

Pokud nastane chyba, je signalizována na výstupy `iE` a v logu. Po fatální chybě se konverze z/do matice přestane provádět. Resetování chyby se pro `mode = 1 .. 4` provede nastavením `LOAD = SAVE = off`, pro `mode = 5 .. 8` resetování fatální chyby nelze provést (musí se přepnout na `mode = 1 .. 4` a pak zpět).

Parametr `nmax` slouží k předalokování výstupního textu. Pokud je `nmax > 0`, naalokuje se při inicializaci úlohy zadané množství znaků a pokud je to nedostatečné, blok hlásí chybu. Pokud je `nmax = 0`, blok zvětšuje délku výstupního textu podle potřeby. Může se zdát nesmyslné zadávat omezení na velikost, když pak může blok selhávat, ale pokud se nezadá, může v extrémním případě dojít k použití celé dostupné paměti, která pak chybí dalším komponentám (např. operačnímu systému) a pak selže celý systém nekontrolovatelně.

Tento blok nepropaguje kvalitu signálu. Více informací je uvedeno v sekci [1.4](#).

Vstup

<code>uMV</code>	Vstupní reference na matici nebo vektor	Reference
<code>uStr</code>	Text pro konverzi na matici/vektor nebo jméno souboru	String
<code>LOAD</code>	Povolení zápisu hodnoty do matice/vektoru	Bool
<code>SAVE</code>	Povolení zápisu hodnoty do souboru/textu	Bool

Parametr

mode	Režim spouštění konverze	⊙2	Long (I32)
	1 aktivované úrovní, do/z souboru		
	2 aktivované hranou, do/z souboru		
	3 aktivované úrovní, do/z textu		
	4 aktivované hranou, do/z textu		
	5 trvale text do matice		
	6 trvale matice do textu		
	7 trvale soubor do matice		
	8 trvale matice do souboru		
format	Formát souboru/textu	⊙1	Long (I32)
	1 CSV		
	2 CSV(středník)		
	3 JSON		
	4 MATLAB		
prec	Počet platných cifer pro každou hodnotu	↓0 ↑20 ⊙6	Long (I32)
TRN	Příznak transpozice matice		Bool
nmax	Rezervovaná paměť pro řetězec	↓0	Long (I32)

Výstup

yMV	Výstupní reference na matici nebo vektor	Reference
yStr	Textová podoba matice/vektoru	String
iE	Kód chyby	Error

MX_VEC – Blok pro uložení dat vektoru

Symbol bloku

Licence: [STANDARD](#)



Popis funkce

Funkční blok **MX_VEC** alokuje paměť (během inicializace bloku) pro n prvků typu určeného parametrem **etype** vektoru referencovaného výstupem **yVec**.

Tento blok nepropaguje kvalitu signálu. Více informací je uvedeno v sekci [1.4](#).

Parametr

n	Počet prvků vektoru	↓1 ↑10000000000 ⊙10	Long (I32)
etype	Typ prvků	⊙8	Long (I32)
	1 Bool		
	2 Byte (U8)		
	3 Short (I16)		
	4 Long (I32)		
	5 Word (U16)		
	6 DWord (U32)		
	7 Float (F32)		
	8 Double (F64)		
	10 Large (I64)		

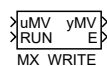
Výstup

yVec	Výstupní reference na vektor	Reference

MX_WRITE – Výpis matice/vektoru do konzole/systémového logu

Symbol bloku

Licence: [STANDARD](#)



Popis funkce

Funkční blok **MX_WRITE** zapíše vektor nebo matici do konzole nebo systémového logu. Závažnost výstupu konzole/systémového logu je nastavena parametrem **mode**. Funkční blok je velmi užitečný pro účely ladění algoritmů s maticemi/vektory.

Výstupní reference **yMV** je vždy nastavena na vstupní referenci **uMV**. Pokud je **RUN = off**, není provedeno nic jiného, jinak je matice nebo vektor zapsán do systémového logu, pokud je cílová úroveň logování pro funkční bloky nastavena na konkrétní **mode**. Formát každého prvku matice/vektoru je určen parametry **mchars** a **mdec**.

Chybový příznak **E** je nastaven na **on**, pokud:

- reference **uMV** není definována (tj. vstup **uMV** není připojen),
- $3 > \text{mchars} > 25$,
- $0 > \text{mdec} > \text{mchars} - 2$.

Tento blok nepropaguje kvalitu signálu. Více informací je uvedeno v sekci [1.4](#).

Vstup

uMV	Vstupní reference na matici nebo vektor	Reference
RUN	Povolení běhu algoritmu	Bool

Parametr

Symbol	Symbolické jméno matice/vektoru pro výstup na konzoli nebo log	String
mchars	Počet znaků jednoho prvku	Long (I32)
mdec	Počet desetinných míst jednoho prvku	Long (I32)

mode	Závažnost výpisu	⊙3 Long (I32)
0	 Žádný	
1	 Žádný	
2	 Podrobný	
3	 Informace	
4	 Varování	
5	 Chyba	

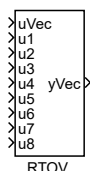
Výstup

yMV	Výstupní reference na matici nebo vektor	Reference
E	Příznak chyby	Bool

RTOV – Vektorový multiplexer

Symbol bloku

Licence: [STANDARD](#)



Popis funkce

Blok **RTOV** slouží k vytvoření vektorových signálů v systému **REXYGEN**. Kombinuje skalární vstupní signály do jednoho výstupního vektorového signálu. Je také možné řetězit bloky **RTOV** pro vytvoření signálů s více než 8 položkami.

Parametr **nmax** definuje maximální počet položek ve vektoru (jinými slovy velikost paměti alokované pro signál). Parametr **offset** definuje pozici prvního vstupního signálu **u1** ve výsledném signálu. Do výsledného vektorového signálu **yVec** jsou kombinovány pouze první **n** vstupních signálů. Parametr **etype** určuje typ vstupních hodnot a typ hodnot ve vnitřním vektoru. Pokud vektor (nebo matice) přivedená na vstup **uVec** má jiný typ, provede se konverze.

POZOR: Do verze 2.50.10.x byl výstupní vektor jednořádkovou maticí. Pozdější verze (2.51.0.9525 a novější) používají jednosloupcovou matici. Tato změna byla nezbytná pro konzistenci v maticových operacích.

Tento blok propaguje kvalitu signálu. Více informací je uvedeno v sekci [1.4](#).

Vstup

uVec	Vektorový signál	Reference
u1..u8	Analogový vstupní signál	Double (F64)

Parametr

nmax	Alokovaná velikost vektoru	↓0 ↻8	Long (I32)
offset	Index prvního vstupu ve vektoru	↓0	Long (I32)
n	Počet použitých vstupů	↓0 ↑8 ↻8	Long (I32)

etype	Typ prvků	⊙8 Long (I32)
	1 Bool	
	2 Byte (U8)	
	3 Short (I16)	
	4 Long (I32)	
	5 Word (U16)	
	6 DWord (U32)	
	7 Float (F32)	
	8 Double (F64)	
	10 Large (I64)	

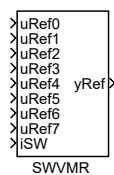
Výstup

yVec	Vektorový signál	Reference

SWVMR – Přepínač vektorového/maticového/odkazovacího signálu

Symbol bloku

Licence: [STANDARD](#)



Popis funkce

Blok **SWVMR** umožňuje přepínání vektorových nebo maticových signálů. Také umožňuje přepínání os pohybu v algoritmech řízení pohybu (viz blok [RM_Axis](#)).

Pro přepínání jednoduchých signálů použijte blok [SSW](#) nebo jeho alternativy [SWR](#) a [SELU](#).

Tento blok nepropaguje kvalitu signálu. Více informací je uvedeno v sekci [1.4](#).

Vstup

<code>uRef0..uRef7</code>	Vektorový signál	Reference
<code>iSW</code>	Selektor aktivního signálu	Long (I32)

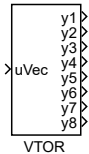
Výstup

<code>yRef</code>	Vektorový signál	Reference

VTOR – Vektorový demultiplexer

Symbol bloku

Licence: [STANDARD](#)



Popis funkce

Blok VTOR rozděljuje vstupní vektorový signál na jednotlivé skalární výstupy. Uživatel definuje počáteční prvek vektoru a počet prvků, které se mají poslat na výstupní signály, pomocí parametrů **offset** a **n**. Dále je možné parametrem **etype** definovat typ výstupních hodnot. Pokud je vektor přivedený na vstup **uVec** jiného typu, provede se konverze.

Tento blok propaguje kvalitu signálu. Více informací je uvedeno v sekci [1.4](#).

Vstup

uVec	Vektorový signál	Reference

Parametr

n	Počet použitých výstupů	↓0 ↑8 ⊙8	Long (I32)
offset	Index prvního výstupu	↓0	Long (I32)
etype	Typ prvků	⊙8	Long (I32)
	1 Bool		
	2 Byte (U8)		
	3 Short (I16)		
	4 Long (I32)		
	5 Word (U16)		
	6 DWord (U32)		
	7 Float (F32)		
	8 Double (F64)		
	10 Large (I64)		

Výstup

y1..y8	Analogový výstupní signál	Double (F64)

Kapitola 15

OPTIM – Bloky pro optimalizaci

Obsah

QP_MPC2QP – Převod úlohy prediktivního řízení na kvadratické programování	558
QP_OASES – Kvadratické programování pomocí metody aktivní množiny	565
QP_UPDATE – Aktualizace matic/vektorů kvadratického programování	570
SOLNP – Solver pro nelineární optimalizaci	575

Knihovna OPTIM je přizpůsobena pro optimalizační algoritmy a procesy. Zahrnuje [QCEDPOPT](#) pro optimalizace tzv. Economic Dispatch problémů, poskytující pokročilé nástroje pro řešení složitých optimalizačních problémů. Knihovna také obsahuje bloky jako [QP_MPC2QP](#) a [QP_OASES](#) pro kvadratické programování, které jsou nezbytné ve scénářích Model Predictive Control (MPC). Navíc je k dispozici [QP_UPDATE](#) pro aktualizaci parametrů kvadratického programu. Tato knihovna je zvláště užitečná v systémech vyžadujících řešení optimalizace na vysoké úrovni, jako jsou pokročilé řídicí a rozhodovací algoritmy.

QP_MPC2QP – Převod úlohy prediktivního řízení na kvadratické programování

Symbol bloku

Licence: [ADVANCED](#)



Popis funkce

Kvadratické programování (Quadratic Programming - QP) je standardní technika, která se velmi dobře hodí k řešení problémů prediktivního řízení založeného na modelu (Model based Predictive Control - MPC) [13]. Kvadratické programování je optimalizační technika, která minimalizuje součet kvadratické formy a lineární formy.

Blok **QP_MPC2QP** převádí lineární problém MPC s kritériem kvadratické optimalizace na problém kvadratického programování. Blok je kompatibilní s blokem [QP_UPDATE](#) a QP řešičem [QP_OASES](#).

Formulace MPC problému

Úloha MPC sestává z diskretního lineárního časově invariantního stavového modelu

$$\begin{aligned} x_{k+1} &= Ax_k + B_u u_k + B_v v_k, \\ y_k &= Cx_k, \end{aligned} \quad (15.1)$$

kde $x \in \mathbb{R}^n$ je stavový vektor, $u \in \mathbb{R}^{m_u}$ je vstupní vektor, $v \in \mathbb{R}^{m_v}$ je vektor poruchy a $y \in \mathbb{R}^p$ je vektor výstupní vektor. Odkazují se na matice $A \in \mathbb{R}^{n \times n}$, $B_u \in \mathbb{R}^{n \times m_u}$, $B_v \in \mathbb{R}^{n \times m_v}$ a $C \in \mathbb{R}^{p \times n}$ pomocí vstupů uA , uBu , uBv a uC . Modelový problém prediktivního řízení je formulován jako optimalizační problém – minimalizace kritéria kvadratické

optimality (cenové funkce) ve tvaru

$$J = \sum_{k=1}^{n_p} \{ \hat{x}_k^T Q \hat{x}_k + \hat{x}_k^T W + \hat{u}_{k-1}^T R \hat{u}_{k-1} \} , \quad (15.2)$$

kde symetrické a kladné (semi-)definitní matice $Q \in \mathbb{R}^{n \times n}$ a $R \in \mathbb{R}^{m_u \times m_u}$ a vektor $W \in \mathbb{R}^n$ jsou odkazovány vstupy $\mathbf{uQ}$, $\mathbf{uR}$ a $\mathbf{uW}$ a n_p je horizont predikce (vstup $\mathbf{np}$).

Pro proces minimalizace mohou být vyžadována další omezení pro stav x a výstup y :

$$x_{\min} \leq x_k \leq x_{\max} \quad (15.3)$$

$$y_{\min} \leq y_k \leq y_{\max} \quad (15.4)$$

Prediktor

Ze stavové rovnice s počáteční podmínkou x_0 platí

$$\begin{aligned} x_1 &= Ax_0 + B_u u_0 + B_v v_0 \\ x_2 &= Ax_1 + B_u u_1 + B_v v_1 \\ &= A^2 x_0 + AB_u u_0 + B_u u_1 + AB_v v_0 + B_v v_1 \\ &\vdots \\ x_k &= A^k x_0 + \begin{bmatrix} A^{k-1} B_u & \dots & AB_u & B_u \end{bmatrix} \begin{bmatrix} u_0 \\ \vdots \\ u_{k-1} \end{bmatrix} \\ &\quad + \begin{bmatrix} A^{k-1} B_v & \dots & AB_v & B_v \end{bmatrix} \begin{bmatrix} v_0 \\ \vdots \\ v_{k-1} \end{bmatrix} \end{aligned}$$

Pro horizont predikce tedy máme n_p

$$\begin{aligned} \underbrace{\begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_{n_p} \end{bmatrix}}_X &= \underbrace{\begin{bmatrix} B_u & 0 & \dots & 0 \\ AB_u & B_u & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ A^{n_p-1} B_u & A^{n_p-2} B_u & \dots & B_u \end{bmatrix}}_{S_u} \underbrace{\begin{bmatrix} u_0 \\ u_1 \\ \vdots \\ u_{n_p-1} \end{bmatrix}}_U \\ &\quad + \underbrace{\begin{bmatrix} B_v & 0 & \dots & 0 \\ AB_v & B_v & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ A^{n_p-1} B_v & A^{n_p-2} B_v & \dots & B_v \end{bmatrix}}_{S_v} \underbrace{\begin{bmatrix} v_0 \\ v_1 \\ \vdots \\ v_{n_p-1} \end{bmatrix}}_V + \underbrace{\begin{bmatrix} A \\ A^2 \\ \vdots \\ A^{n_p} \end{bmatrix}}_T x_0 \end{aligned}$$

i.e.

$$X = S_u U + S_v V + T x_0 \quad (15.5)$$

Podobně můžeme získat výstupní rovnici

$$\begin{aligned} \underbrace{\begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_{n_p} \end{bmatrix}}_Y &= \begin{bmatrix} Cx_1 \\ Cx_2 \\ \vdots \\ Cx_{n_p} \end{bmatrix} = \underbrace{\begin{bmatrix} CB_u & 0 & \dots & 0 \\ CAB_u & CB_u & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ CA^{n_p-1}B_u & CA^{n_p-2}B_u & \dots & CB_u \end{bmatrix}}_{S_{cu}} \begin{bmatrix} u_0 \\ u_1 \\ \vdots \\ u_{n_p-1} \end{bmatrix} \\ &+ \underbrace{\begin{bmatrix} CB_v & 0 & \dots & 0 \\ CAB_v & CB_v & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ CA^{n_p-1}B_v & CA^{n_p-2}B_v & \dots & CB_v \end{bmatrix}}_{S_{cv}} \begin{bmatrix} v_0 \\ v_1 \\ \vdots \\ v_{n_p-1} \end{bmatrix} + \underbrace{\begin{bmatrix} CA \\ CA^2 \\ \vdots \\ CA^{n_p} \end{bmatrix}}_{T_c} x_0 \end{aligned}$$

tedy

$$Y = S_{cu}U + S_{cv}V + T_c x_0 \quad (15.6)$$

a standardní QP matice A_{eq} a b_{eq}

$$A_{eq} = S_{cu}L, \quad b_{eq} = -S_{cv}V - T_c x_0 \quad (15.7)$$

Prediktor pro horizont kontroly menší než horizont predikce

Doposud se předpokládalo, že se bude hledat optimální řízení v celém horizontu predikce n_p . Pro dlouhý predikční horizont to vede k časově náročné optimalizaci, kterou lze urychlit volbou kontrolního horizontu n_c (vstup `nc`) menšího, než je predikční horizont n_p . Potom lze U zapsat jako

$$U = \left[\begin{array}{c} u_0 \\ u_1 \\ \vdots \\ u_{n_c-1} \\ u_{n_c-1} \\ \vdots \\ u_{n_p-1} \end{array} \right] \left. \begin{array}{l} \left. \vphantom{\begin{matrix} u_0 \\ u_1 \\ \vdots \\ u_{n_c-1} \end{matrix}} \right\} n_c \\ \left. \vphantom{\begin{matrix} u_{n_c-1} \\ u_{n_c-1} \\ \vdots \\ u_{n_p-1} \end{matrix}} \right\} n_p - n_c \end{array} \right\} n_p$$

Všimněte si, že vstup u_k je rozdíl stavu a pro řídicí horizont n_c platí $u_{k+n_c-1} = u_{k+n_c} =$

$u_{k+n_c+1} = \dots = u_{k+n_p-1} = 0$ (pro krok k). Pak to lze napsat jako

$$U = \begin{matrix} n_c \\ \vdots \\ n_p - n_c \end{matrix} \left\{ \begin{matrix} \begin{bmatrix} I_{m_u} & & \mathbf{0} \\ & \ddots & \\ \mathbf{0} & & I_{m_u} \\ & & & \mathbf{0} \end{bmatrix} \\ \underbrace{\hspace{10em}}_L \end{matrix} \right\} \begin{bmatrix} u_0 \\ \vdots \\ u_{n_c-1} \end{bmatrix} \triangleq LU_{n_c} \quad (15.8)$$

kde $U \in \mathbb{R}^{n_p \cdot m_u}$, $U_{n_c} \in \mathbb{R}^{n_c \cdot m_u}$ a $I_{m_u} \in \mathbb{R}^{m_u \times m_u}$ je matice identity.

Rovnice (15.5) a (15.6) jsou upraveny pro U_{n_c} na

$$X = S_u LU_{n_c} + S_v V + T x_0 \quad (15.9)$$

$$Y = S_{cu} LU_{n_c} + S_{cv} V + T_c x_0 \quad (15.10)$$

Matice $S_u L \in \mathbb{R}^{n_p \cdot n \times n_c \cdot m_u}$, $S_v \in \mathbb{R}^{n_p \cdot n \times n_p \cdot m_v}$, $T \in \mathbb{R}^{n_p \cdot n \times n}$, $S_{cu} L \in \mathbb{R}^{n_p \cdot p \times n_c \cdot m_u}$, $S_{cv} \in \mathbb{R}^{n_p \cdot p \times n_p \cdot m_v}$ a $T_c \in \mathbb{R}^{n_p \cdot p \times n}$ jsou počítány tímto blokem, musí být přiděleny např. bloky **MX_MAT** a odkazy na předem alokované matice musí být připojeny k blokovým vstupům **uSuL**, **uSv**, **uT**, **uScuL**, **uScv** a **uTc**.

Výchozí hodnota matice $L \in \mathbb{R}^{n_p \cdot m_u \times n_c \cdot m_u}$ v rovnici 15.8 vybírá první n_c subvektory $u_i, i = 0, \dots, n_c - 1$ od U . Blok také umožňuje vybrat n_c subvektory u_i s libovolnými indexy od $0, \dots, n_p$, které jsou obsaženy v celočíselném vektoru dimenze n_c , na kterou odkazuje vstup **uL**. Prvky tohoto vektoru musí tvořit rostoucí sekvenci. Pokud není připojen vstup **uL**, použije se výchozí hodnota L (stejná hodnota L se získá, pokud je vektor, na který odkazuje **uL** roven $[0, 1, \dots, n_c - 1]^T$).

Konverze MPC se stejnými predikčními a kontrolními horizonty na QP

Standardní forma cenové funkce QP je

$$J_{QP} = hU^T H U + U^T G \quad (15.11)$$

kde U je vektor optimální řídicí sekvence, H je symetrická a kladná (semi-) definitní Hessova matice, G je gradientový vektor a h je skalární konstanta, která je obvykle rovná 1 nebo 1/2.

Cenovou funkci (15.2) lze upravit na formu

$$\begin{aligned}
J &= \sum_{k=1}^{n_p} \{x_k^T Q x_k + x_k^T W + u_{k-1}^T R u_{k-1}\} \\
&= [x_1 \quad x_2 \quad \dots \quad x_{n_p}] \underbrace{\begin{bmatrix} Q & & \\ & Q & \\ & & \ddots \\ & & & Q \end{bmatrix}}_{\bar{Q} = \bar{Q}^T} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_{n_p} \end{bmatrix} + [x_1 \quad x_2 \quad \dots \quad x_{n_p}] \underbrace{\begin{bmatrix} W \\ W \\ \vdots \\ W \end{bmatrix}}_{\bar{W}} \\
&\quad + [u_0 \quad u_1 \quad \dots \quad u_{n_p-1}] \underbrace{\begin{bmatrix} R & & \\ & R & \\ & & \ddots \\ & & & R \end{bmatrix}}_{\bar{R} = \bar{R}^T} \begin{bmatrix} u_0 \\ u_1 \\ \vdots \\ u_{n_p-1} \end{bmatrix} \\
&= X^T \bar{Q} X + X^T \bar{W} + U^T \bar{R} U \\
&= (U^T S_u^T + V^T S_v^T + x_0^T T^T) \bar{Q} (S_u U + S_v V + T x_0) \\
&\quad + (U^T S_u^T + V^T S_v^T + x_0^T T^T) \bar{W} + U^T \bar{R} U \\
&= U^T S_u^T \bar{Q} S_u U + U^T S_u^T \bar{Q} S_v V + U^T S_u^T \bar{Q} T x_0 \\
&\quad + V^T S_v^T \bar{Q} S_u U + V^T S_v^T \bar{Q} S_v V + V^T S_v^T \bar{Q} T x_0 \\
&\quad + x_0^T T^T \bar{Q} S_u U + x_0^T T^T \bar{Q} S_v V + x_0^T T^T \bar{Q} T x_0 \\
&\quad + U^T S_u^T \bar{W} + V^T S_v^T \bar{W} + x_0^T T^T \bar{W} + U^T \bar{R} U \\
&= U^T (S_u^T \bar{Q} S_u + \bar{R}) U + U^T S_u^T (2\bar{Q} S_v V + 2\bar{Q} T x_0 + \bar{W}) \\
&\quad + \underbrace{V^T S_v^T (\bar{Q} S_v V + 2\bar{Q} T x_0 + \bar{W}) + x_0^T T^T (\bar{Q} T x_0 + \bar{W})}_{J_{\text{dif}}} \triangleq J_{QP} + J_{\text{dif}}
\end{aligned} \tag{15.12}$$

kde J_{dif} je konstanta nezávislá na U . Odtud plyne

$$J_{QP} = U^T (S_u^T \bar{Q} S_u + \bar{R}) U + U^T S_u^T (2\bar{Q} S_v V + 2\bar{Q} T x_0 + \bar{W}) \tag{15.13}$$

Při porovnání této rovnice s (15.11) je zřejmé, že

$$\begin{aligned}
H &= \frac{1}{h} (S_u^T \bar{Q} S_u + \bar{R}) \\
G &= S_u^T (2\bar{Q} S_v V + 2\bar{Q} T x_0 + \bar{W})
\end{aligned} \tag{15.14}$$

Konverze MPC s horizontem řízení menším než horizont predikce

Podobně jako v předchozí podsekcí můžeme dostat za $n_c < n_p$

$$\begin{aligned} H &= \frac{1}{h} L^T (S_u^T \bar{Q} S_u + \bar{R}) L \\ G &= (S_u L)^T (2 \bar{Q} S_v V + 2 \bar{Q} T x_0 + \bar{W}) \triangleq G_v V + G_x x_0 + G_w \end{aligned} \quad (15.15)$$

kde matice L je definována pomocí (15.8). Hessova matice H je konstantní matice pro všechny kroky k MPC. Ale gradientový vektor G se obecně mění v každém kroku k , protože vektory V a x_0 se mění. Proto se G skládá z částí G_v , G_x a G_w , které jsou již konstantními vektory. Matice H a vektory G_v , G_x a G_w jsou vypočítány tímto funkčním blokem a jsou odkazovány vstupy `uH`, `uGv`, `uGx` a `uGw`, které musí být předem přiděleno. Skalární konstanta h je parametr funkčního bloku.

Tento blok nepropaguje kvalitu signálu. Více informací je uvedeno v sekci 1.4.

Vstup

<code>np</code>	Horizont predikce	↓1 ↑1000000	Long (I32)
<code>nc</code>	Horizont řízení	↓1 ↑1000000	Long (I32)
<code>uA</code>	Vstupní odkaz na matici systému A		Reference
<code>uBu</code>	Vstupní odkaz na matici Bu vektoru řízení u		Reference
<code>uBv</code>	Vstupní odkaz na matici Bv vektoru poruchy v		Reference
<code>uC</code>	Vstupní odkaz na výstupní matici C		Reference
<code>uQ</code>	Vstupní odkaz na symetrickou matici Q v kritériu optimality		Reference
<code>uW</code>	Vstupní odkaz na vektor W v kritériu optimality		Reference
<code>uR</code>	Vstupní odkaz na symetrickou matici R v kritériu optimality		Reference
<code>uI</code>	Vstupní odkaz na celočíselný vektor indexů I		Reference
<code>uH</code>	Vstupní odkaz na Hessovu matici H		Reference
<code>uGx</code>	Vstupní odkaz na složku gradientního vektoru G odpovídající stavu x		Reference
<code>uGv</code>	Vstupní odkaz na složku gradientního vektoru G odpovídající poruše v		Reference
<code>uGw</code>	Vstupní odkaz na složku gradientního vektoru G odpovídající vektoru W		Reference
<code>uSuL</code>	Vstupní odkaz na pracovní matici Su*L		Reference
<code>uSv</code>	Vstupní odkaz na pracovní matici Sv		Reference
<code>uT</code>	Vstupní odkaz na pracovní matici T		Reference
<code>uScuL</code>	Vstupní odkaz na pracovní matici Scu*L		Reference
<code>uScv</code>	Vstupní odkaz na pracovní matici Scv		Reference
<code>uTc</code>	Vstupní odkaz na pracovní matici Tc		Reference
<code>uWORK</code>	Vstupní odkaz na pracovní matici WORK		Reference
<code>HLD</code>	Pozastavení		Bool

Výstup

<code>yA</code>	Výstupní odkaz na matici systému A	Reference

yBu	Výstupní odkaz na matici Bu vektoru řízení u	Reference
yBv	Výstupní odkaz na matici Bv vektoru poruchy v	Reference
yC	Výstupní odkaz na výstupní matici C	Reference
yQ	Výstupní odkaz na symetrickou matici Q v kritériu optimality	Reference
yW	Výstupní odkaz na vektor W v kritériu optimality	Reference
yR	Výstupní odkaz na symetrickou matici R v kritériu optimality	Reference
yl	Výstupní odkaz na celočíselný vektor indexů l	Reference
yH	Výstupní odkaz na Hessovu matici H	Reference
yGx	Výstupní odkaz na složku gradientního vektoru G odpovídající stavu x	Reference
yGv	Výstupní odkaz na složku gradientního vektoru G odpovídající poruše v	Reference
yGw	Výstupní odkaz na složku gradientního vektoru G odpovídající vektoru W	Reference
ySuL	Výstupní odkaz na pracovní matici Su*L	Reference
ySv	Výstupní odkaz na pracovní matici Sv	Reference
yT	Výstupní odkaz na pracovní matici T	Reference
yScuL	Výstupní odkaz na pracovní matici Scu*L	Reference
yScv	Výstupní odkaz na pracovní matici Scv	Reference
yTc	Výstupní odkaz na pracovní matici Tc	Reference
yWORK	Výstupní odkaz na pracovní matici WORK	Reference
E	Příznak chyby	Bool

QP_OASES – Kvadratické programování pomocí metody aktivní množiny

Symbol bloku

Licence: [ADVANCED](#)

>uQP	yQP>
>uH	yH>
>uG	yG>
>uA	yA>
>uLB	yLB>
>uUB	yUB>
>uLBA	yLBA>
>uUBA	yUBA>
>uXopt	yXopt>
>uYopt	yYopt>
>unWSR	ynWSR>
>utime	ytime>
>VAR	objval>
>INIT	E>
>HLD	IE>
QP_OASES	

Popis funkce

Blok QP_OASES řeší kvadratický program pomocí metody aktivní množiny [14]:

$$\begin{aligned} \min_{x \in \mathbb{R}^{nV}} \quad & \frac{1}{2}x^T H x + x^T G, \\ \text{s. t.} \quad & lbA \leq Ax \leq ubA, \\ & lb \leq x \leq ub, \end{aligned}$$

kde nV je počet proměnných, nC je počet omezení, Hessova matice $H \in \mathbb{R}^{nV \times nV}$ je symetrická a pozitivně (semi-)definitní, vektor gradientu $G \in \mathbb{R}^{nV}$, matice omezení $A \in \mathbb{R}^{nC \times nV}$, vektory omezujících hodnot $lb, ub \in \mathbb{R}^{nV}$ a vektory omezujících hodnot $lbA, ubA \in \mathbb{R}^{nC}$.

Blok obaluje knihovnu [qpOASES](#)¹, jejíž použití je popsáno v manuálu [15].

Výstupní reference yH , yG , yA , yLB , yUB , $yLBA$, $yUBA$, $yXopt$ a $yYopt$ jsou vždy nastaveny na odpovídající vstup uH , uG , uA , uLB , uUB , $uLBA$, $uUBA$, $uXopt$ a $uYopt$. Pokud není připojeno uQP , konkrétní kvadratický problém (QP) je alokovan při prvním spuštění funkčního bloku (viz níže) a výstup yQP je nastaven na referenci alokovaného QP. Pokud je uQP připojeno (na výstup yQP předchozího bloku QP_OASES), výstup yQP je nastaven na uQP a blok pracuje s již alokovaným QP.

Blok používá interní proměnné nV a nC . Hodnota nV je nastavena na počet řádků vektoru G referencovaného uG , hodnota nV je nastavena na počet řádků matice A refe-

¹[qpOASES](#), která je distribuována pod licencí GNU Lesser General Public License, viz Příloha A k [15].

rencované `uA`. Pokud reference `uA` není definována (matice A není připojena), hodnota `nC = 0`.

Pro řešení QP problému je v obecném případě vytvořen objekt `QProblem` (viz Kapitola 3 [15]). Blok může také řešit následující speciální případy v závislosti na vstupních referencích a parametru `hessianType`:

uH není připojeno. V tomto případě se předpokládá, že Hessova matice má triviální hodnotu identitní nebo nulovou matici. Parametr `hessianType` musí být roven `HST_ZERO` nebo `HST_IDENTITY`, viz Sekce 4.5 manuálu [15].

uA není připojeno. V tomto případě se matice omezení A nepoužívá (`nC = 0`), je vytvořen objekt `QProblemB`, viz Sekce 4.3 manuálu [15]. Parametr `hessianType` může být libovolná povolená hodnota.

VAR = on. Pokud je vstup `VAR = on` během prvního spuštění bloku, je vytvořen objekt třídy `SQProblem`, viz Sekce 4.2 manuálu [15]. V tomto případě mohou všechny vstupní matice a vektory měnit v každém kroku provedení, když `VAR = on`.

Pro získání řešení QP problému musí být alespoň jedna z vstupních referencí `uXopt` a `uYopt` definována (připojena k vektoru). Pokud je připojeno k `uXopt`, výstup `yXopt` bude odkazovat na primární řešení $Xopt \in \mathbb{R}^{nV}$, pokud je připojeno k `uYopt`, výstup `yYopt` bude odkazovat na duální řešení $Yopt \in \mathbb{R}^{nV+nC}$ QP problému. Pokud jsou oba vstupy připojeny, obě řešení budou získána v každém kroku. Optimální hodnota objektivní funkce je uvedena na výstupu `objval`.

Celočíselný vstup `unWSR` určuje maximální počet rekalkulací pracovní sady, které mají být provedeny během počáteční homotopie, viz Sekce 3.2 manuálu [15]. Výstup `ynWSR` obsahuje skutečný počet provedených rekalkulací pracovní sady. Pokud je připojen dvojitý vstup `utime` a má kladnou hodnotu, obsahuje maximální povolený čas CPU v sekundách pro celou inicializaci. Skutečný požadovaný čas CPU pro inicializaci je uveden na výstupu `ytime`.

Alespoň jeden vektor musí být připojen k vstupu `uXopt` nebo `uYopt` a oba musí být připojeny k získání řešení QP problému. Pokud je připojeno `uXopt`, výstup `yXopt` bude odkazovat na primární řešení $Xopt$, pokud je připojeno `uYopt`, výstup `yYopt` bude odkazovat na duální řešení $Yopt$ úlohy QP. Pokud jsou oba vstupy připojeny, obě řešení budou získána v každém kroku.

Pokud je vstup `INIT = on`, pak je konkrétní alokovaný QP problém znovu inicializován. `INIT` by mělo být `on` po dobu jediného období (hrany), protože během inicializace QP není vypočteno žádné řešení. Pokud je `HLD = on`, pak se nic nepočítá.

Chybový příznak `E` je nastaven na `on` a chybový kód `iE` je nastaven na nulu, pokud:

- reference `uG` nebo `uLB` nebo `uUB` není definována (tj. vstup `uG` nebo `uLB` nebo `uUB` není připojen),
- reference `uA` je definována a `uLBA` nebo `uUBA` není definována (tj. vstup `uA` je připojen a `uLBA` nebo `uUBA` není připojena),

- obě reference `uXopt` a `uYopt` nejsou definovány (tj. není připojen žádný z vstupů `uXopt` a `uYopt`),
- Hessova matice H referencovaná `uH` má odlišný počet řádků a sloupců než nV ,
- počet řádků vektorů referencovaných `uLB` a `uUB` není roven nV (nebo počet jejich sloupců není roven 1),
- počet řádků vektorů referencovaných `uLBA` a `uUBA` není roven nC (nebo počet jejich sloupců není roven 1) pokud je matice A referencovaná `uA` připojena,
- počet řádků vektoru referencovaného `uXopt` není roven nV nebo počet řádků vektoru referencovaného `yOpt` není roven $nV+nC$ (nebo počet jejich sloupců není roven 1),
- interní prostor pro transponované kopie matic H nebo A je příliš malý.

Pokud je příznak `E` nastaven na `on` a chybový kód `iE` není nula, pak `iE` označuje chybový kód `qpOASES`, viz soubor `MessageHandling.hpp` z knihovny `qpOASES`.

Tento blok nepropaguje kvalitu signálu. Více informací je uvedeno v sekci 1.4.

Vstup

<code>uQP</code>	Vstupní odkaz na úlohu kvadratického programování	Reference
<code>uH</code>	Vstupní odkaz na Hessovu matici H	Reference
<code>uG</code>	Vstupní odkaz na gradientní vektor G	Reference
<code>uA</code>	Vstupní odkaz na matici omezení A	Reference
<code>uLB</code>	Vstupní odkaz na vektor dolních mezí LB	Reference
<code>uUB</code>	Vstupní odkaz na vektor horních mezí UB	Reference
<code>uLBA</code>	Vstupní odkaz na vektor dolních mezí LBA omezujících podmínek	Reference
<code>uUBA</code>	Vstupní odkaz na vektor horních mezí UBA omezujících podmínek	Reference
<code>uXopt</code>	Vstupní odkaz na vektor prvotního optimálního řešení	Reference
<code>uYopt</code>	Vstupní odkaz na vektor duálního optimálního řešení	Reference
<code>unWSR</code>	Maximální počet výpočtů pracovní množiny během inicializace	Long (I32)
<code>utime</code>	Maximální dovolený čas CPU v sekundách pro celou inicializaci	Double (F64)
<code>VAR</code>	Příznak časové variantnosti matic H a A	Bool
<code>INIT</code>	Při každém spuštění bloku volá funkci <code>init()</code> místo <code>hotstart()</code>	Bool
<code>HLD</code>	Pozastavení	Bool

Parametr

<code>nVmax</code>	Maximální počet optimalizačních proměnných nV	Long (I32)
<code>nCmax</code>	Maximální počet optimalizačních omezujících podmínek nC	Long (I32)

<code>hessianType</code>	Typ Hessianovy matice	Long (I32)
<code>printLevel</code>	Úroveň výpisů	Long (I32)
<code>enableRamping</code>	Povolit tzv. rampování	Bool
<code>enableFarBounds</code>	Povolit použití tzv. dalekých mezí	Bool
<code>enableFlippingBounds</code>	Povolit používání tzv. převrácených mezí (flipping bounds)	Bool
<code>enableRegularisation</code>	Povolit regularizaci semidefinitní Hessianovy matice	Bool
<code>enableFullLITests</code>	Povolit použití tvrdších testů lineární nezávislosti	Bool
<code>enableNZCTests</code>	Povolit testy nenulového zakřivení	Bool
<code>enableDriftCorrection</code>	Korekce frekvence nebo driftu (0 = off)	Long (I32)
<code>enableCholeskyRefact</code>	Frekvence plné refaktorizace projektovaného Hessiánu (0 = off)	Long (I32)
<code>enableEqualities</code>	Rovnosti se vždy považují za aktivní omezení	Bool
<code>terminationTolerance</code>	Ukončovací tolerance	Double (F64)
<code>boundTolerance</code>	Pokud se horní a spodní meze liší méně než o tuto toleranci, jsou považovány za shodné, tj. za podmínku s rovností	Double (F64)
<code>boundRelaxation</code>	Počáteční relaxace mezí pro startovací homotopii a počáteční hodnota dalekých mezí	Double (F64)
<code>epsNum</code>	Tolerance čitatele pro poměrové testy	Double (F64)
<code>epsDen</code>	Tolerance jmenovatele pro poměrové testy	Double (F64)
<code>maxPrimalJump</code>	Maximálně povolený skok v primálních proměnných pro testy lineární nezávislosti	Double (F64)
<code>maxDualJump</code>	Maximálně povolený skok v duálních proměnných pro testy lineární nezávislosti	Double (F64)
<code>initialRamping</code>	Počáteční hodnota pro strategii rampování	Double (F64)
<code>finalRamping</code>	Koncová hodnota pro strategii rampování	Double (F64)
<code>initialFarBounds</code>	Počáteční velikost dalekých mezí	Double (F64)
<code>growFarBounds</code>	Faktor zvětšování dalekých mezí	Double (F64)
<code>initialStatusBounds</code>	Počáteční stav mezí v první iteraci	Long (I32)
<code>epsFlipping</code>	Tolerance kvadrátu prvku na Choleského diagonále, která aktivuje tzv. převrácené meze (flipping bounds)	Double (F64)
<code>numRegularisationSteps</code>	Maximální počet po sobě jdoucích regularizačních kroků	Long (I32)
<code>epsRegularisation</code>	Měřítkovací faktor identické matice použitý pro regularizaci Hessiánu	Double (F64)
<code>numRefinementSteps</code>	Maximální počet kroků pro iteračních upřesnění	Long (I32)
<code>epsIterRef</code>	Tolerance předčasného ukončení pro iterativní upřesnění	Double (F64)
<code>epsLITests</code>	Tolerance pro testy lineární nezávislosti	Double (F64)
<code>epsNZCTests</code>	Tolerance pro testy nenulového zakřivení	Double (F64)

Výstup

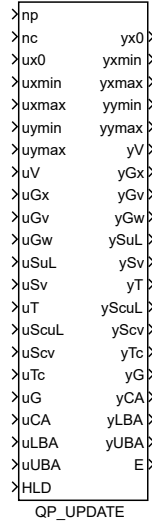
<code>yQP</code>	Výstupní odkaz na úlohu kvadratického programování	Reference
<code>yH</code>	Výstupní odkaz na Hessianovu matici H	Reference
<code>yG</code>	Výstupní odkaz na gradientní vektor G	Reference

yA	Výstupní odkaz na matici omezení A	Reference
yLB	Výstupní odkaz na vektor dolních mezí LB	Reference
yUB	Výstupní odkaz na vektor horních mezí UB	Reference
yLBA	Výstupní odkaz na vektor dolních mezí LBA omezujících podmínek	Reference
yUBA	Výstupní odkaz na vektor horních mezí UBA omezujících podmínek	Reference
yXopt	Výstupní odkaz na vektor prvotního optimálního řešení	Reference
yYopt	Výstupní odkaz na vektor duálního optimálního řešení	Reference
ynWSR	Skutečný počet výpočtů pracovní množiny během inicializace	Long (I32)
ytime	Spotřebovaný čas CPU v sekundách pro celou inicializaci	Double (F64)
objval	Optimální hodnota účelové funkce	Double (F64)
E	Příznak chyby	Bool
iE	Kód chyby	Long (I32)

QP_UPDATE – Aktualizace matic/vektorů kvadratického programování

Symbol bloku

Licence: [ADVANCED](#)



Popis funkce

Blok QP_UPDATE spolupracuje s blokem QP_MPC2QP, který převádí problém MPC popsany rovnicemi (15.1)–(15.4) s predikčním horizontem n_p a řídicím horizontem n_c (vstupy `np` a `nc`), do kvadratického programování a předpočítá Hessiany matice H , části gradientového vektoru G_x , G_v , G_w , matice určující stavová omezení S_uL , S_v , T a matice určující výstupní omezení $S_{cu}L$, S_{cv} , T_c . Kromě konstantní Hessovské matice H jsou ostatní vektory a matice napojeny na vstupní reference `uGx`, `uGv`, `uGw`, `uSuL`, `uSv`, `codeuT`, `uScuL`, `uScv` a `uTc`.

Tento blok aktualizuje QP problém pro daný časový okamžik s aktuálními hodnotami počáteční podmínky stavového vektoru x_0 , omezení stavového vektoru $x_{\min}$ a $x_{\max}$, omezení výstupního vektoru $y_{\min}$ a $y_{\max}$, vektoru V (viz rovnice (15.5)) predikčních vektorů rušení v_k , $k = 0, \dots, n_p - 1$. Tyto vektory jsou napojeny na vstupní reference `ux0`, `uxmin`, `uxmax`, `uymin`, `ymax` a `uV`.

Nejprve je gradientový vektor G napojený na vstup `uG` aktualizován podle rovnice (15.15):

$$G = G_x x_0 + G_v V + G_w.$$

Stavová omezení (15.3) mohou být přepsána pomocí (15.9) pro predikční horizont n_p na

$$X_{\min} - S_v V - T x_0 \leq S_u L U_{n_c} \leq X_{\max} - S_v V - T x_0,$$

kde $X_{\min}$ resp. $X_{\max}$ je vektor složený z n_p kopií vektoru $x_{\min}$ resp. $x_{\max}$. Obdobně mohou být výstupní omezení (15.4) přepsána pomocí (15.10) na

$$Y_{\min} - S_{cv}V - T_c x_0 \leq S_{cu} L U_{n_c} \leq Y_{\max} - S_{cv}V - T_c x_0, .$$

kde $Y_{\min}$ resp. $Y_{\max}$ je vektor složený z n_p kopií vektoru $y_{\min}$ resp. $y_{\max}$.

Kompaktnější forma těchto dvou rovnic je

$$\underbrace{\begin{bmatrix} X_{\min} - S_v V - T x_0 \\ Y_{\min} - S_{cv} V - T_c x_0 \end{bmatrix}}_{\text{lb}A} \leq \underbrace{\begin{bmatrix} S_u L \\ S_{cu} L \end{bmatrix}}_{CA} U_{n_c} \leq \underbrace{\begin{bmatrix} X_{\max} - S_v V - T x_0 \\ Y_{\max} - S_{cv} V - T_c x_0 \end{bmatrix}}_{\text{ub}A}, \quad (15.16)$$

kde matice CA a vektory lbA, ubA jsou vypočítány tímto blokem, musí být alokovány např. bloky `MX_MAT` a reference na předalokované matice musí být napojeny na vstupní bloky uCA, uLBA a uUBA.

Poslední rovnice 15.16 je obecná forma QP omezení. Pokrývá jak rovnostní, tak nerovnostní omezení pro stavy a výstupy.

Pokud nejsou požadována žádná stavová omezení, nechte vstupy `uxmin`, `uxmax`, `uSuL`, `uSv` a `uT` nepřipojené. Potom rovnice 15.16 dostane tvar

$$\underbrace{[X_{\min} - S_v V - T x_0]}_{\text{lb}A} \leq \underbrace{[S_u L]}_{CA} U_{n_c} \leq \underbrace{[X_{\max} - S_v V - T x_0]}_{\text{ub}A}. \quad (15.17)$$

Podobně pokud nejsou požadována žádná výstupní omezení, nechte vstupy `uymin`, `uymax`, `uScuL`, `uScv` a `uTc` nepřipojené. Potom rovnice 15.16 dostane tvar

$$\underbrace{[Y_{\min} - S_{cv} V - T_c x_0]}_{\text{lb}A} \leq \underbrace{[S_{cu} L]}_{CA} U_{n_c} \leq \underbrace{[Y_{\max} - S_{cv} V - T_c x_0]}_{\text{ub}A}. \quad (15.18)$$

Výstupní reference `yx0`, `yxmin`, `yxmax`, `yymin`, `ymax`, `yV`, `yGx`, `yGv`, `yGw`, `ySuL`, `ySv`, `yT`, `yScuL`, `yScv`, `yTc`, `yG`, `yCA`, `yLBA` and `yUBA` are always set to the corresponding input `ux0`, `uxmin`, `uxmax`, `uymin`, `uymax`, `uV`, `uGx`, `uGv`, `uGw`, `uSuL`, `uSv`, `uT`, `uScuL`, `uScv`, `uTc`, `uG`, `uCA`, `uLBA` and `uUBA`.

Pokud `HLD = on`, pak se nic nepočítá.

Chybový příznak `E` je nastaven na `on`, pokud:

- horizont predikce `np < 1` nebo kontrolní horizont `nc < 1`, nebo `nc > np`,
- reference `ux0` není definována nebo typ prvku pole, na které odkazuje, není `Double` (F64),
- interní proměnná `bStateConstr = on` a alespoň jedna z referencí `uxmin`, `uxmax` není definována, nebo typ prvku alespoň jednoho pole, na které odkazuje, není `Double` (F64),

- interní proměnná `bOutputConstr = on` a reference `uymin` je definována a typ prvku pole, na které odkazuje, není `Double` (F64),
- interní proměnná `bOutputConstr = on` a reference `uymax` je definována a typ prvku pole, na které odkazuje, není `Double` (F64),
- reference `uV` je definována a typ prvku pole, na které odkazuje, není `Double` (F64),
- reference `uG` je definována a alespoň jedna z referencí `uGx`, `uGv`, `uSuL`, `uSv` nebo `uT` není definována,
- reference `uG` je definována a typ prvku pole, na které odkazuje, není `Double` (F64), nebo reference `uGx` je definována a typ prvku pole, na které odkazuje, není `Double` (F64), nebo reference `uGv` je definována a typ prvku pole, na které odkazuje, není `Double` (F64), nebo reference `uGw` je definována a typ prvku pole, na které odkazuje, není `Double` (F64),
- reference `uSuL` je definována a typ prvku pole, na které odkazuje, není `Double` (F64), nebo reference `uSv` je definována a typ prvku pole, na které odkazuje, není `Double` (F64), nebo reference `uT` je definována a typ prvku pole, na které odkazuje, není `Double` (F64),
- reference `uScuL` je definována a typ prvku pole, na které odkazuje, není `Double` (F64), nebo reference `uScv` je definována a typ prvku pole, na které odkazuje, není `Double` (F64), nebo reference `uTc` je definována a typ prvku pole, na které odkazuje, není `Double` (F64),
- reference `uCA` nebo `uLBA` nebo `uUBA` nebo typ prvku alespoň jednoho z polí, na které odkazují, není `Double` (F64),
- pole, na která odkazují definované odkazy, jsou příliš malá nebo mají nekompatibilní rozměry.

Pokud `E = on`, podrobnosti naleznete v systémovém protokolu.

Tento blok nepropaguje kvalitu signálu. Více informací je uvedeno v sekci 1.4.

Vstup

<code>np</code>	Horizont predikce	↓1 ↑1000000	Long (I32)
<code>nc</code>	Horizont řízení	↓1 ↑1000000	Long (I32)
<code>ux0</code>	Vstupní odkaz na vektor počátečních podmínek <code>x0</code> vektoru stavu <code>x</code>		Reference
<code>uxmin</code>	Vstupní odkaz na vektor spodních mezí prvků stavového vektoru		Reference
<code>uxmax</code>	Vstupní odkaz na vektor horních mezí prvků stavového vektoru		Reference
<code>uymin</code>	Vstupní odkaz na vektor spodních mezí výstupních nerovností		Reference

uymax	Vstupní odkaz na vektor horních mezí výstupních nerovností	Reference
uV	Vstupní odkaz na vektor predikovaných poruch	Reference
uGx	Vstupní odkaz na složku gradientního vektoru G odpovídající stavu x	Reference
uGv	Vstupní odkaz na složku gradientního vektoru G odpovídající poruše v	Reference
uGw	Vstupní odkaz na složku gradientního vektoru G odpovídající vektoru W	Reference
uSuL	Vstupní odkaz na pracovní matici Su^*L	Reference
uSv	Vstupní odkaz na pracovní matici Sv	Reference
uT	Vstupní odkaz na pracovní matici T	Reference
uScuL	Vstupní odkaz na pracovní matici Scu^*L	Reference
uScv	Vstupní odkaz na pracovní matici Scv	Reference
uTc	Vstupní odkaz na pracovní matici Tc	Reference
uG	Vstupní odkaz na gradientní vektor G	Reference
uCA	Vstupní odkaz na matici omezení kvadratického programování CA	Reference
uLBA	Vstupní odkaz na vektor dolních mezí LBA omezujících podmínek	Reference
uUBA	Vstupní odkaz na vektor horních mezí UBA omezujících podmínek	Reference
HLD	Pozastavení	Bool

Výstup

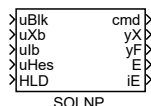
yx0	Výstupní odkaz na vektor počátečních podmínek x0 vektoru stavu x	Reference
yxmin	Výstupní odkaz na vektor spodních mezí prvků stavového vektoru	Reference
yxmax	Výstupní odkaz na vektor horních mezí prvků stavového vektoru	Reference
yymin	Výstupní odkaz na vektor spodních mezí výstupních nerovností	Reference
yymax	Výstupní odkaz na vektor horních mezí výstupních nerovností	Reference
yV	Výstupní odkaz na vektor predikovaných poruch	Reference
yGx	Výstupní odkaz na složku gradientního vektoru G odpovídající stavu x	Reference
yGv	Výstupní odkaz na složku gradientního vektoru G odpovídající poruše v	Reference
yGw	Výstupní odkaz na složku gradientního vektoru G odpovídající vektoru W	Reference
ySuL	Výstupní odkaz na pracovní matici Su^*L	Reference
ySv	Výstupní odkaz na pracovní matici Sv	Reference
yT	Výstupní odkaz na pracovní matici T	Reference
yScuL	Výstupní odkaz na pracovní matici Scu^*L	Reference

yScv	Výstupní odkaz na pracovní matici Scv	Reference
yTc	Výstupní odkaz na pracovní matici Tc	Reference
yG	Výstupní odkaz na gradientní vektor G	Reference
yCA	Výstupní odkaz na matici omezení kvadratického programování CA	Reference
yLBA	Výstupní odkaz na vektor dolních mezí LBA omezujících podmínek	Reference
yUBA	Výstupní odkaz na vektor horních mezí UBA omezujících podmínek	Reference
E	Příznak chyby	Bool

SOLNP – Solver pro nelineární optimalizaci

Symbol bloku

Licence: [MATRIX](#)



Popis funkce

Blok **SOLNP** je navržen k řešení obecného problému nelineárního programování (NLP) ve tvaru

$$\min f(x)$$

s omezeními

$$\begin{aligned} g(x) &= 0, \\ l_h &\leq h(x) \leq u_h, \\ l_x &\leq x \leq u_x, \end{aligned}$$

kde

- $x \in \mathbb{R}^n$ je vektor návrhových proměnných, které mají být optimalizovány,
- $f(x)$ je ztrátová funkce (objektivní funkce),
- $g(x)$ je vektor rovnostních omezení,
- $h(x)$ je vektor nerovnostních omezení,
- l_h je vektor dolních mezí nerovnostních omezení,
- u_h je vektor horních mezí nerovnostních omezení,
- l_x je vektor dolních mezí návrhových proměnných,
- u_x je vektor horních mezí návrhových proměnných.

Řešič **SOLNP** se snaží najít optimální řešení $x^* = \arg \min_x f(x)$ s využitím metody rozšířeného Lagrangiánu a sekvenčního kvadratického programování (SQP).

Technické detaily lze nalézt v [16] a [17]. Obecně jsou f , g a h libovolné nelineární hladké funkce.

Co se týče implementace, ztrátová funkce, rovnostní a nerovnostní omezení mohou být definovány odděleně, například prostřednictvím bloku **REXLANG**, který umožňuje definici vlastních funkcí. Jeho výstup **y0** je poté zpětně předán do vstupu **uBlk** bloku **SOLNP**.

Při formulaci optimalizačního problému, který má být vyřešen blokem **SOLNP**, mohou být použity následující pomocné proměnné:

- Vektor počátečních odhadů: x_0 .
- Matice x_b definující rozsah návrhových proměnných $x_b = [l_x, u_x]$.
- Matice uIb definující nerovnostní omezení: $uIb = [l_h, u_h]$ (volitelně, použito pouze pokud optimalizační problém obsahuje nerovnostní omezení).

Blok může být nakonfigurován třemi způsoby na základě vstupního vektoru uXb :

1. $uXb = [x_0, x_b]$: řešič použije zadaný počáteční odhad i meze pro návrhové proměnné.
2. $uXb = x_0$: řešič použije pouze zadaný počáteční odhad.
3. $uXb = x_b$: řešič použije pouze zadané meze pro návrhové proměnné. Počáteční odhad je automaticky vypočten jako $x_0 = \frac{l_x + u_x}{2}$.

Vektor uXb může mít jeden, dva nebo tři sloupce; a počet řádků odpovídá počtu návrhových proměnných (rozměr x).

Tento blok nepropaguje kvalitu signálu. Více informací je uvedeno v sekci 1.4.

Vstup

uBlk	Odkaz na spolupracující blok REXLANG	Reference
uXb	Počáteční odhad a meze pro návrhové proměnné	Reference
uIb	Meze pro nerovnostní omezení (volitelně)	Reference
uHes	Uživatelsky dodaná Hessova matice (volitelně)	Reference
HLD	Pozastavení	Bool

Parametr

rho	Penalizační parametr pro ztrátovou funkci definovanou pomocí rozšířeného Lagrangianu	Double (F64)
		⊙1.0
MaxIterMajor	Maximální počet hlavních iterací	Long (I32)
		↓0 ⊙400
MaxIterMinor	Maximální počet vedlejších iterací	Long (I32)
		↓0 ⊙200
delta	Parametr perturbace pro numerický výpočet gradientu	Long (I32)
		↓1.79769e+308 ↑double
tolerance	Parametr tolerance pro splnitelnost a optimálnost	Long (I32)
		↓1.79769e+308 ↑double
nmax	Rezervovaná paměť pro pole	Long (I32)
		↓10 ↑100000 ⊙100

Výstup

cmd	Číslo fáze/příkazu pro spolupracující blok REXLANG	Long (I32)
yX	Vstupní vektor pro spolupracující blok REXLANG	Reference
yF	Výstupní vektor pro spolupracující blok REXLANG	Reference
E	Příznak chyby	Bool

iE

Kód chyby

Long (I32)

Kapitola 16

SPEC – Speciální bloky

Obsah

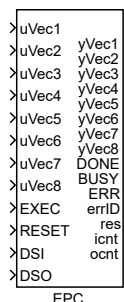
EPC – Spouštění externích programů	580
HTTP – Blok pro generování požadavků HTTP GET a POST (za- staralý)	583
HTTP2 – Blok pro generování požadavků HTTP	585
RDC – Komunikační blok	587
REXLANG – Volně programovatelný blok	590
SMTP – Blok pro odesílání e-mailových oznámení přes SMTP . .	611
STEAM – Přepočet vlastností páry	613
UART – Komunikační blok pro UART	616

Knihovna SPEC zahrnuje rozmanitou sadu funkčních bloků navržených pro integraci široké škály funkcí do automatizace, řídicích systémů a komunikačních protokolů. Od usnadnění přesných termodynamických výpočtů s blokem [STEAM](#) po umožnění bezproblémové datové komunikace prostřednictvím [UART](#) a [SMTP](#) slouží knihovna jako komplexní sada nástrojů pro inženýry a vývojáře. Obsahuje specializované bloky pro spouštění externích programů ([EPC](#)), zpracování webových požadavků ([HTTP2](#)). Navíc nabízí unikátní vstupně-výstupní řešení ([RDC](#)) a všestranné programovací prostředí s [REXLANG](#).

EPC – Spouštění externích programů

Symbol bloku

Licence: [ADVANCED](#)



Popis funkce

Blok **EPC** vykoná externí program, když je detekována náběžná hrana (**off**→**on**) na vstupu **EXEC**. Název programu a parametry jsou specifikovány parametrem **cmd**, formátované přesně tak, jak by byly na příkazové řádce operačního systému.

Data mohou být předávána mezi systémem **REXYGEN** a externím programem pomocí souborů. Formát souboru, specifikovaný parametrem **format**, je textový, aby byla zajištěna kompatibilita a snadné použití s různými softwarovými nástroji. Například data lze načíst v **MATLABu** pomocí příkazu:

```
values = load('-ASCII', 'epc_inputVec1');
```

a v **SCILABu** s:

```
values = read('epc_inputVec1', -1, 32);
```

Upravte název souboru a počet sloupců podle potřeby vašeho konkrétního projektu. Data lze také získat zpět do systému **REXYGEN** pomocí stejného formátu souboru.

Blok **EPC** pracuje ve dvou režimech:

Základní režim: Spuštěný náběžnou hranou na **EXEC**, tento režim čte aktuální vstupní data a ukládá je do určených souborů specifikovaných parametrem **ifns**. Každý *i*-tý vstupní vektor **uVec<i>** je uložen do svého odpovídajícího *i*-tého souboru.

Vzorkovací režim: V tomto režimu jsou data z vstupních vektorů neustále ukládána do souborů v každém algoritmickém cyklu, což usnadňuje záznam dat v reálném čase.

V obou režimech jsou data z jednoho časového okamžiku uložena v jednom řádku v souboru. Obdobně jsou data z výstupních souborů získána a přiřazena výstupům bloku (**yVec<i>**), přičemž každý řádek z *i*-tého výstupního souboru jde do *i*-tého výstupního vektoru.

Vstupy pracující ve *vzorkovacím režimu* jsou definovány seznamem `sl`¹. Výstupy pracují vždy ve *vzorkovacím režimu* – poslední hodnoty jsou udržovány, když je dosaženo konce souboru. Kopírování dat do vstupních souborů může být blokováno vstupem `DSI`, to samé platí pro výstupní data a vstup `DSO`.

Pro sloučení více signálů do jednoho vektoru použijte blok `RTOV`. Pro rozložení vektoru na jednotlivé signály použijte blok `VTOR`. Tyto bloky lze řetězit pro zpracování vektorů libovolné velikosti, což zvyšuje flexibilitu bloku při zpracování složitých datových struktur.

Poznámky:

- Volaný externí program má stejnou prioritu jako volající úloha. Tato priorita je vysoká, v některých případech vyšší než úlohy jádra operačního systému. Na systémech založených na Linuxu je možné snížit prioritu pomocí příkazu `chrt`:
`chrt -o 0 extprg.sh,`
 kde `extprg.sh` je původní externí program.
- Velikost signálů je omezena parametrem `nmax`. Větší parametr znamená větší spotřebu paměti, takže zvolte tento parametr co nejmenší.
- Názvy souborů musí respektovat konvence pojmenování operačního systému cílové platformy. Doporučuje se používat pouze alfanumerické znaky a podtržítko, aby se předešlo problémům. Dodržujte také velká a malá písmena.
- Blok také vytváří kopie souborů `ifns` a `ofns` z implementačních důvodů. Názvy těchto souborů jsou rozšířeny o podtržítko.
- Cesty `ifns` a `ofns` jsou relativní k adresáři, kde jsou uloženy archivy systému REXYGEN. Doporučuje se definovat symbolický odkaz na RAM-disk uvnitř tohoto adresáře pro zlepšení výkonu. Na druhou stranu, pro dlouhé řady dat je lepší ukládat data na trvalé úložiště, protože data lze např. po obnovení po výpadku napájení přidat.
- Blok `OSCALL` lze také použít pro provedení některých funkcí operačního systému.

Tento blok nepropaguje kvalitu signálu. Více informací je uvedeno v sekci 1.4.

Vstup

<code>uVec1..uVec8</code>	Vstupní vektorový signál	Reference
<code>EXEC</code>	Náběžná hrana spouští externí program	<code>Bool</code>
<code>RESET</code>	Reset bloku	<code>Bool</code>
<code>DSI</code>	Pozastavení vzorkování na vstupech	<code>Bool</code>
<code>DSO</code>	Pozastavení vzorkování na výstupech	<code>Bool</code>

¹Např. 1,3..5,8. Programy třetích stran (Simulink, OPC klienti, atd.) pracují s celým číslem, které reprezentuje binární masku, např. 157 (binárně 10011101) v uvedeném příkladu.

Parametr

cmd	Externí program		String
ifns	Vstupní soubory (oddělené středníkem)	⊙epc_uVec1;epc_uVec2	String
ofns	Výstupní soubory (oddělené středníkem)	⊙epc_yVec1;epc_yVec2	String
sl	Seznam vzorkovacích vstupů (oddělené čárkou)	↓0 ↑255 ⊙85	Long (I32)
ifm	Maximální počet vzorků ve vstupním souboru	⊙10000	Long (I32)
format	Formát vstupních a výstupních souborů	⊙1	Long (I32)
	1 textový (pouze hodnoty)		
	2 CSV (desetinná tečka a čárky)		
	3 CSV (desetinná čárka a středníky)		
nmax	Maximální délka výstupních vektorů	↓2 ↑1000000 ⊙100	Long (I32)

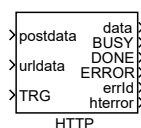
Výstup

yVec1..yVec8	Výstupní vektorový signál	Reference
DONE	Příznak skončení externího programu	Bool
BUSY	Příznak běhu externího programu	Bool
ERR	Příznak chyby	Bool
errID	Kód chyby	Error
res	Návratový kód externího programu	Long (I32)
icnt	Aktuální číslo vzorku na vstupech	Long (I32)
ocnt	Aktuální číslo vzorku na výstupech	Long (I32)

HTTP – Blok pro generování požadavků HTTP GET a POST (zastaralý)

Symbol bloku

Licence: [ADVANCED](#)



Popis funkce

Blok HTTP je zastaralý, nahraďte ho blokem [HTTP2](#).

Blok HTTP vykoná jednorázový HTTP požadavek. Cílová adresa (URL) je definována parametrem `url` a vstupem `urldata`. Konečná URL je vytvořena tak, že vstup `urldata` je jednoduše přidán k parametru `url`.

HTTP požadavek je spuštěn na náběžnou hranu (`off`→`on`) na vstupu `TRG`. Poté je výstup `BUSY` nastaven na `on` dokud není požadavek dokončen, což je signalizováno výstupem `DONE=on`. V případě chyby je výstup `ERROR` nastaven. Výstup `errId` nese poslední chybu identifikovanou systémovým kódem chyby `REXYGEN`. Výstup `hterror` nese kód stavu HTTP. Všechna data poslaná serverem klientovi jsou uložena ve výstupu `data`.

Blok může být spuštěn v blokovacím nebo neblokovacím režimu, což je specifikováno parametrem `BLOCKING`. V blokovacím režimu je vykonání úlohy pozastaveno, dokud požadavek neskončí. V neblokovacím režimu blok vykoná pouze jednu operaci v závislosti na dostupných datech a vykonání úlohy není blokováno. Doporučuje se vždy spouštět blok HTTP v neblokovacím režimu. Je však nutné zmínit, že na různých operačních systémech některé operace nelze provést v neblokovacím režimu, takže buďte opatrní a nepoužívejte tento blok v rychlých úlohách ([QTASK](#)) nebo v úlohách s krátkou dobou vykonání. Nejlepší podpora neblokovacího provozu je na operačním systému GNU/Linux. Maximální doba trvání požadavku provedeného blokem HTTP je specifikována parametrem `timeout`.

Blok podporuje ověření uživatele pomocí základní metody ověřování HTTP. Uživatelské jméno a heslo lze specifikovat parametry `user` a `password`. Blok také podporuje zabezpečený HTTP (HTTPS). Je také možné nechat blok ověřit certifikát serveru nastavením parametru `VERIFY`. SSL certifikát serveru nebo důvěryhodný certifikační orgán serveru musí být uložen v parametru `certificate` ve formátu PEM. Blok nepodporuje žádné uložení certifikátů.

Parametry `postmime` a `acceptmime` specifikují MIME kódování dat odesílaných na server nebo očekávané kódování HTTP odpovědi.

Parametry `nmax`, `postmax` a `datamax` specifikují maximální velikosti bufferů alokovaných blokem. Parametr `nmax` je maximální velikost libovolného řetězce. Parametr

`postmax` specifikuje maximální velikost `postdata`. Parametr `datamax` specifikuje maximální velikost `data`.

Tento blok nepropaguje kvalitu signálu. Více informací je uvedeno v sekci 1.4.

Vstup

<code>postdata</code>	Data vložená do HTTP požadavku	String
<code>urldata</code>	Data připojená k URL adrese	String
<code>TRG</code>	Spuštění zvolené akce	Bool

Parametr

<code>url</code>	URL adresa pro odeslání HTTP požadavku	String
<code>method</code>	Typ HTTP požadavku	⊙1 Long (I32)
	1 GET	
	2 POST	
	3 PUT	
	4 DELETE	
	5 HEAD	
	6 TRACE	
	7 PATCH	
	8 OPTIONS	
	9 CONNECT	
<code>user</code>	Uživatelské jméno	String
<code>password</code>	Heslo	String
<code>certificate</code>	Certifikát pro autentizaci	String
<code>VERIFY</code>	Povolení verifikace serveru (platnost certifikátu)	Bool
<code>postmime</code>	Typ kódování pro požadavek POST	⊙application/json String
<code>acceptmime</code>	Typ kódování pro požadavek GET	⊙application/json String
<code>timeout</code>	Povolená doba pro dokončení operace	⊙5.0 Double (F64)
<code>BLOCKING</code>	Čekání na dokončení operace	Bool
<code>nmax</code>	Rezervovaná paměť pro řetězec	↓0 ↑65520 Long (I32)
<code>postmax</code>	postmax	↓128 ↑65520 ⊙256 Long (I32)
<code>datamax</code>	Alokovaná paměť pro odpověď na příkaz HTTP	↓128 ↑10000000 ⊙1024 Long (I32)

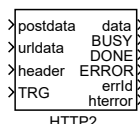
Výstup

<code>data</code>	Data z odpovědi	String
<code>BUSY</code>	Odesílání HTTP požadavku	Bool
<code>DONE</code>	HTTP požadavek byl zpracován	Bool
<code>ERROR</code>	Příznak chyby	Bool
<code>errId</code>	Kód chyby	Error
<code>hterror</code>	HTTP odpověď	Long (I32)

HTTP2 – Blok pro generování požadavků HTTP

Symbol bloku

Licence: [ADVANCED](#)



Popis funkce

Blok HTTP2 vykoná jednorázový HTTP požadavek. Cílová adresa (URL) je definována parametrem `url` a vstupem `urldata`. Konečná URL je vytvořena tak, že vstup `urldata` je přidán k parametru `url`. Vstup `header` může být použit pro deklaraci dalších hlavičkových polí.

HTTP požadavek je spuštěn na náběžnou hranu (`off`→`on`) na vstupu `TRG`. Poté je výstup `BUSY` nastaven na `on` dokud není požadavek dokončen, což je signalizováno výstupem `DONE=on`. V případě chyby se na výstupu `ERROR` nastaví hodnota `on`. Výstup `errId` nese poslední chybu identifikovanou systémovým kódem chyby `REXYGEN`. Výstup `hterror` zobrazuje kód stavu HTTP. Všechna data poslaná serverem klientovi jsou uložena ve výstupu `data`. Všechny chybové výstupy jsou resetovány, když je nový HTTP požadavek spuštěn na vstupu `TRG`.

Blok může být spuštěn v blokovacím nebo neblokovacím režimu, což je specifikováno parametrem `BLOCKING`. V blokovacím režimu je vykonání úlohy pozastaveno, dokud požadavek neskončí. V neblokovacím režimu blok vykoná pouze jednu operaci v závislosti na dostupných datech a vykonání úlohy není blokováno. Doporučuje se vždy spouštět blok HTTP v neblokovacím režimu. Je však nutné zmínit, že na různých operačních systémech některé operace nelze provést v neblokovacím režimu, takže buďte opatrní a nepoužívejte tento blok v rychlých úlohách (`QTASK`) nebo v úlohách s krátkou dobou vykonání. Nejlepší podpora neblokovacího provozu je na operačním systému GNU/Linux. Maximální doba trvání požadavku provedeného blokem HTTP je specifikována parametrem `timeout`.

Blok podporuje ověření uživatele pomocí základní metody ověřování HTTP. Uživatelské jméno a heslo lze specifikovat parametry `user` a `password`. Blok také podporuje zabezpečený HTTP (HTTPS). Je také možné nechat blok ověřit certifikát serveru nastavením parametru `VERIFY`. SSL certifikát serveru nebo důvěryhodný certifikační orgán serveru musí být uložen v parametru `certificate` ve formátu PEM. Blok nepodporuje žádné uložení certifikátů.

Parametry `postmime` a `acceptmime` specifikují MIME kódování dat odesílaných na server nebo očekávané kódování HTTP odpovědi.

Parametry `nmax`, `postmax` a `datamax` specifikují maximální velikosti bufferů alokovaných blokem. Parametr `nmax` je maximální velikost libovolného řetězce. Parametr `postmax` specifikuje maximální velikost `postdata`. Parametr `datamax` specifikuje maxi-

mální velikost **data**.

Tento blok nepropaguje kvalitu signálu. Více informací je uvedeno v sekci 1.4.

Vstup

postdata	Data vložená do HTTP požadavku	String
urldata	Data připojená k URL adrese	String
header	Uživatelské položky hlavičky	String
TRG	Spuštění zvolené akce	Bool

Parametr

url	URL adresa pro odeslání HTTP požadavku	String
method	Typ HTTP požadavku	⊙1 Long (I32)
	1 GET	
	2 POST	
	3 PUT	
	4 DELETE	
	5 HEAD	
	6 TRACE	
	7 PATCH	
	8 OPTIONS	
	9 CONNECT	
user	Uživatelské jméno	String
password	Heslo	String
certificate	Certifikát pro autentizaci	String
VERIFY	Povolení verifikace serveru (platnost certifikátu)	Bool
postmime	Typ kódování pro požadavek POST	⊙application/json String
acceptmime	Typ kódování pro požadavek GET	⊙application/json String
timeout	Povolená doba pro dokončení operace	⊙5.0 Double (F64)
BLOCKING	Čekání na dokončení operace	Bool
nmax	Rezervovaná paměť pro řetězec	↓0 ↑65520 Long (I32)
postmax	postmax	↓128 ↑65520 ⊙4096 Long (I32)
datamax	Alokovaná paměť pro odpověď na příkaz HTTP	↓128 ↑10000000 ⊙64000 Long (I32)

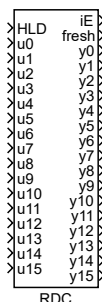
Výstup

data	Data z odpovědi	String
BUSY	Odesílání HTTP požadavku	Bool
DONE	HTTP požadavek byl zpracován	Bool
ERROR	Příznak chyby	Bool
errId	Kód chyby	Error
hterror	HTTP odpověď	Long (I32)

RDC – Komunikační blok

Symbol bloku

Licence: [ADVANCED](#)



Popis funkce

Blok **RDC** je speciální vstupně-výstupní blok. Hodnoty se předávají mezi dvěma bloky se stejným číslem, ale na různých počítačích (popřípadě na stejném počítači mezi dvěma Simulinky nebo Simulinkem a systémem **REXYGEN**). Hodnoty se předávají UDP/IP protokolem. Komunikace tedy funguje na všech lokálních sítích LAN i na linkách sítě Internet. Algoritmus v každém kroku provádí následující operace:

- Pokud je vstup **HLD=on**, je vykonávání bloku pozastaveno.
- Má-li parametr **period** kladnou hodnotu, je vyhodnocen rozdíl mezi systémovým časovačem a časem posledního odeslaného paketu. Vykonávání bloku je zastaveno, pokud rozdíl nepřesahuje hodnotu parametru **period**. Je-li parametr **period** nulový nebo záporný, rozdíl časů se nekontroluje.
- Vytvoří se datový paket. Paket obsahuje číslo bloku, tzv. **invoke** číslo (pořadové číslo paketu) a hodnoty **u0** až **u15**. Všechny hodnoty jsou uloženy ve standardním tzv. síťovém bytovém pořadí, takže aplikace je nezávislá na počítači a/nebo procesoru.
- Paket je odeslán na zadanou IP adresu a port.
- Číslo **invoke** je zvětšeno o 1.
- Je zkontrolováno, zda byly přijaty nějaké příchozí pakety.
- Pokud ano, je zkontrolována platnost paketu (velikost, číslo **id**, **invoke** číslo).
- Pokud je paket v pořádku, jsou všechny výstupy **y0** až **y15** nastaveny na hodnoty obsažené v přijatém paketu.

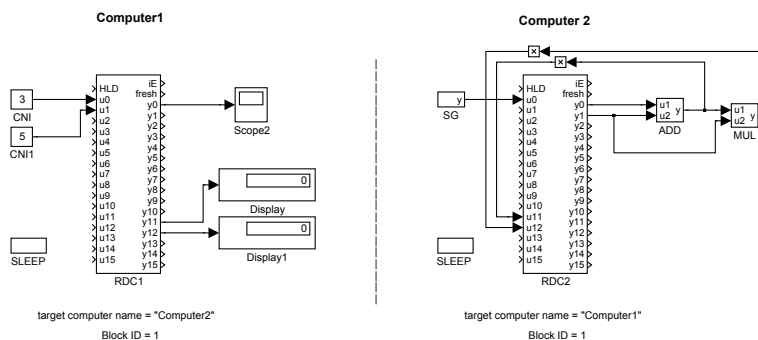
- Je aktualizován výstup **fresh**. V případě chyby je chybový kód zobrazen na výstup **err**.

V každém směru je mezi dvěma bloky se stejným **id** číslem periodicky přenášeno 16 hodnot. Vstup **u(i)** prvního bloku je přenesen na výstup **y(i)** druhého bloku. Na rozdíl od TCP/IP protokolu nemá UDP/IP protokol žádný mechanismus pro zacházení se ztracenými nebo duplicitními pakety, takže je třeba se s tím vypořádat algoritmem samotným. Pro tento účel slouží **invoke** číslo. Tato stavová proměnná je zvětšena o 1 pokaždé, když je odeslán paket. Blok také uchovává **invoke** číslo posledního přijatého paketu. Je možné rozlišit různé události porovnáním těchto dvou čísel. Pakety s **invoke** čísly menšími než **invoke** číslo posledního přijatého paketu jsou zamítnuty, pokud rozdíl není větší než 10. Tím je vyřešena situace, kdy je jeden z bloků RDC restartován a s ním jeho **invoke** číslo.

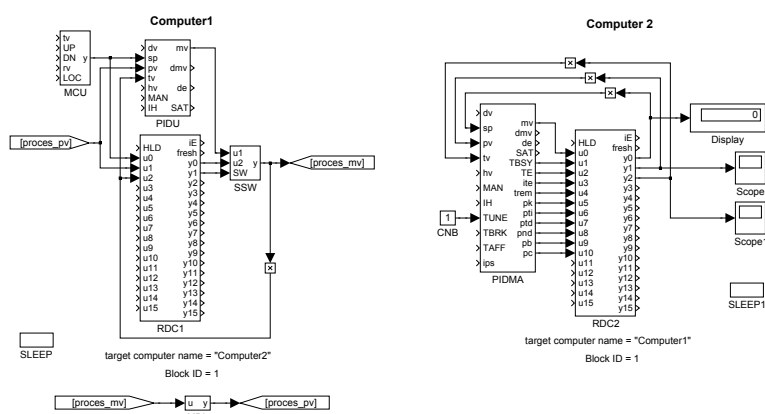
Poznámka: Všechny bloky RDC ve stejné aplikaci musí mít stejný **local port** a počet bloků RDC je z implementačních důvodů omezen na 64. Pokud běží na stejném počítači dvě aplikace používající blok RDC, musí každá z nich používat jiný **local port**.

Příklady

Konstanty 3 a 5 jsou odesílány z Počítače 1 na Počítač 2, kde se objeví na výstupech **y0** a **y1** bloku RDC2. Konstanty jsou sečteny, vynásobeny a odeslány zpátky na Počítač 1 prostřednictvím vstupů **u11** a **u12** bloku RDC2. Zobrazení připojená k výstupům **y11** a **y12** bloku RDC1 zobrazují výsledky matematických operací $3 + 5$ a $(3 + 5) * 5$. Signál z generátoru **SG** běžícího na Počítači 2 je přenášén na výstup **y0** bloku RDC1, kde může být snadno zobrazen.



Jednoduchost příkladu je záměrná. Cílem je demonstrovat funkčnost bloku, nikoliv složitost systému. Ve skutečnosti je blok RDC používán v komplexnějších úkolech, například pro vzdálené ladění PID regulátoru, jak je ukázáno níže. PID algoritmus běží na Počítači 1, zatímco ladící algoritmus je prováděn na Počítači 2. Viz bloky **PIDU**, **PIDMA** a **SSW** pro více informací.



Tento blok nepropaguje kvalitu signálu. Více informací je uvedeno v sekci 1.4.

Vstup

HLD	Pozastavení	Bool
u0..u15	Signál k odeslání do vzdáleného RDC bloku	Double (F64)

Parametr

target	Jméno/IP adresa vzdáleného počítače	String
rport	Port vzdáleného počítače	⊙1288 Word (U16)
lport	Lokální port	⊙1288 Word (U16)
id	Identifikátor bloku	↓1 ↑32767 ⊙1 Long (I32)
period	Minimální perioda komunikace [s]	Double (F64)

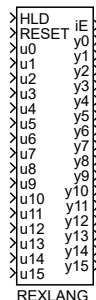
Výstup

iE	Kód chyby	Long (I32)
	0 bez chyby	
	-1 překročen maximální počet bloků (>64)	
	-2 konflikt lokálních portů	
	-3 nelze otevřít socket	
	-4 nelze přiřadit lokální port	
	-5 nelze nastavit tzv. neblokující mód socketu	
	-6 chyba v IP adrese	
	-10 ... chyba v inicializaci socketové knihovny	
	1 bez chyby, ale zatím nepřišla žádná data	
	2 přijat neplatný paket	
	4 chyba při příjmu paketu	
	8 chyba při odesílání paketu	
fresh	Čas uplynulý od příjmu posledního paketu [s]	Double (F64)
y0..y15	Signál přijatý ze vzdáleného RDC bloku	Double (F64)

REXLANG – Volně programovatelný blok

Symbol bloku

Licence: [REXLANG](#)



Popis funkce

Systém REXYGEN obsahuje řadu standardních funkčních bloků, které splňují běžné požadavky řídicích aplikací. V některých případech je však nutné vytvořit vlastní blok, který bude provádět specifické úkoly. Pro tento účel je k dispozici skriptovací jazyk REXLANG. Tento blok umožňuje tvorbu uživatelsky definovaných algoritmů se způsobem kódování specifikovaným parametrem **src type**. Dostupné možnosti kódování zahrnují:

- **1: C-like** – Tento skriptovací jazyk je podobný C (nebo Java). Kód je ukládán v textovém souboru s příponou **.c**. Jazyk dodržuje níže uvedenou syntaxi podobnou C.
- **2: STL** – Kód je ukládán do textového souboru s příponou **.stl**. Kód sleduje standard **IEC61131-3**, který je implementován s určitými omezeními podobnými těm v C-like skriptu, jako jsou žádné struktury a omezené typy dat (INT, REAL a STRING). Proměnné pro funkční bloky jsou deklarovány jako globální proměnné v **VAR_INPUT**, výstupy jako **VAR_OUTPUT** a parametry jako **VAR_PARAMETER**. Zahrnuje standardní funkce jak jsou specifikovány, s systémovými a komunikačními funkcemi identickými s těmi v možnosti C-like.
- **3: RLB** – Binární soubor vytvořený kompilací skriptů C-like nebo STL. Tato možnost je vhodná pro uživatele, kteří nechtějí zveřejňovat zdrojový kód svého bloku.
- **4: ILL** – Textový soubor obsahující mnemokódy podobné instrukcím assembleru. Tato možnost v současné době není podporována.

Skriptovací jazyk (C-like)

Skriptovací jazyk použitý v REXLANG se podobá jazyku C, ale existují významné rozdíly a omezení:

- Podporované datové typy zahrnují `double`, `long` a `string`. Typy jako `int`, `short` a `bool` jsou interně konvertovány na `long`. Typ `float` je konvertován na `double`. Typ `typedef` není dostupný.
- Ukazatele a struktury nejsou implementovány. Pole definovat lze a indexace je možná pomocí operátoru `[ ]`. Vstupy, výstupy a parametry bloku nemohou být pole.
- Operátor `' , '` není implementován.
- Preprocesor podporuje příkazy jako `#include`, `#define`, `#ifdef .. [#else ..] #endif`, and `#ifndef .. [#else ..] #endif`. Nicméně `#pragma` a `#if .. [#else ..] #endif` nejsou podporovány.
- Standardní knihovny ANSI C nejsou implementovány, ale mnoho matematických funkcí z `math.h` a další funkce jsou dostupné (viz níže).
- Klíčová slova `input`, `output` a `parameter` se používají pro odkazování na vstupy, výstupy a parametry bloku `REXLANG`. Jsou implementovány systémové funkce pro řízení běhu a diagnostiku.
- Funkce `main()` se provádí periodicky během běhu. K dispozici jsou i další funkce jako `init()` (prováděná jednou při spuštění), `exit()` (prováděná při zastavení řídicího algoritmu) a `parchange()` (prováděná při změnách parametrů).
- Ve funkcích a procedurách bez parametrů musí být v deklaraci explicitně uvedeno `void`.
- Přetěžování identifikátorů není povoleno, tj. klíčová slova a vestavěné funkce nemohou sdílet názvy s identifikátory. Lokální a globální proměnné musí mít různé názvy.
- Uživatelsky definované návratové hodnoty z `main()`, `init()` a `exit()` jsou zapsány do výstupu `iE`. Hodnoty `< -99` zastaví provádění algoritmu (vyžaduje vstup `RESET` pro další provádění). Návratové hodnoty jsou kategorizovány následovně:

<code>iE >= 0</code>	indikuje, že vykonávání je bez chyb.
<code>0 > iE >= -99</code>	znamená varování bez ovlivnění provádění funkčního bloku.
<code>iE < -99</code>	naznačuje chybu, která zastaví provádění funkčního bloku.

Syntaxe skriptovacího jazyka

Syntaxe skriptovacího jazyka je založena na jazyku C, ale s některými úpravami:

- `<typ> input(<číslo vstupu>) <název proměnné>;` pro vstupní proměnné.
- `<typ> output(<číslo výstupu>) <název proměnné>;` pro výstupní proměnné.
- `<typ> parameter(<číslo parametru>) <název proměnné>;` pro parametrické proměnné.

Proměnné `input` a `parameter` jsou pouze pro čtení, zatímco do proměnných `output` lze číst i zapisovat.

Příklad:

```
double input(1) vstup; // Deklarace proměnné vstup typu double, která představuje hodnotu vstupu bloku u1.
long output(2) vystup; // Deklarace proměnné vystup typu long, která představuje hodnotu výstupu bloku y2.

vstup = 3; // Nedovolený příkaz - do vstupu nelze přiřazovat.
if (vstup > 1) vystup = 3 + vstup; // Správné použití.
```

Dostupné funkce

Skriptovací jazyk obsahuje širokou škálu funkcí, včetně matematických výpočtů, operací s vektory, manipulace s řetězci a systémových příkazů. Funkce jsou kategorizovány a podrobně popsány následovně:

- **Matematické funkce** (v souladu s knihovnou `math.h` ANSI C):

Tato kategorie zahrnuje funkce jako `atan`, `sin`, `cos`, `exp`, `log`, `sqrt`, `tan`, `asin`, `acos`, `fabs`, `fmod`, `sinh`, `cosh`, `tanh`, `pow`, `atan2`, `ceil`, `floor` a `abs`. Všimněte si, že `abs` je speciálně navržen pro celočíselné hodnoty, zatímco ostatní operují s proměnnými typu `double`. Pro výpočet absolutní hodnoty desetinného čísla slouží funkce `fabs`.

- **Vektorové funkce** (nejsou v ANSI C) :

Tato sada obsahuje specializované funkce pro práci s vektory, které jsou reprezentovány jako pole hodnot. Příklady inicializace polí lze nalézt na konci dokumentace bloku. Mezi funkce této kategorie patří:

```
double max([n,] val1, ..., valn)
    Vrací maximální hodnotu mezi zadanými prvky. První parametr, označující počet prvků, je volitelný.

double max(n, vec)
    Hledá maximální hodnotu ve vektoru vec.

double min([n,] val1, ..., valn)
    Podobně jako max, ale vrací minimální hodnotu.

double min(n, vec)
    Hledá minimální hodnotu ve vektoru vec.

double poly([n,] x, an, ..., a1, a0)
    Vypočítá hodnotu polynomu  $y = anx^n + \dots + a1x + a0$ . První parametr je volitelný.

double poly(n, x, vec)
    Vypočítá hodnotu polynomu  $y = vec[n]x^n + \dots + vec[1]x + vec[0]$ .

double scal(n, vec1, vec2)
    Vypočítá skalární součin dvou vektorů:  $y = vec1[0]vec2[0] + \dots + vec1[n-1]vec2[n-1]$ .
```

`double scal(n, vec1, vec2, skip1, skip2)`

Varianta `scal`, která umožňuje přeskočit prvky: $y = \text{vec1}[0]\text{vec2}[0] + \text{vec1}[\text{skip1}]\text{vec2}[\text{skip2}] + \dots + \text{vec1}[(n-1)\text{skip1}]\text{vec2}[(n-1)\text{skip2}]$. Tato varianta je vhodná pro násobení matic, které jsou uloženy jako vektory (řádek po řádku nebo sloupec po sloupci).

`double conv(n, vec1, vec2)`

Vypočítá konvolutorní součin dvou vektorů: $y = \text{vec1}[0]\text{vec2}[n-1] + \text{vec1}[1]\text{vec2}[n-2] + \dots + \text{vec1}[n-1]\text{vec2}[0]$.

`double sum(n, vec)`

Sečte prvky vektoru: $y = \text{vec}[0] + \text{vec}[1] + \dots + \text{vec}[n-1]$.

`double sum([n,] val1, ..., valn)`

Sečte zadané hodnoty. Počet prvků je volitelný.

`[] subarray(idx, vec)`

Vrátí pole `vec` začínající od indexu `idx`. Typ vrácené hodnoty je automaticky vybrán podle pole `vec`.

`copyarray(count, vecSource, idxSource, vecTarget, idxTarget)`

Kopíruje `count` prvků pole `vecSource`, začínající od indexu `idxSource`, do pole `vecTarget`, začínající od indexu `idxTarget`. Obě pole musí být stejného typu.

`void fillarray(vector, value, count)`

Kopíruje hodnotu `value` do `count` prvků pole `vector` (vždy od indexu 0).

Poznámka: Funkce `max`, `min`, `poly`, `scal`, `conv` a `sum` jsou přetížené, což znamená, že mají několik variant založených na parametrech. Parametry jsou přísně kontrolovány podle typu, vyžadující přetypování pro typy jiné než `double`. Například:

```
double res = max(dVal, (double)iVal, 1.0, (double)2);
```

přetypuje `iVal` na `double`. Pokud parametr není typu `double`, je hlášena chyba s textem "no function of this prototype was found".

- **Funkce pro práci s řetězci** (Tato sekce pokrývá funkce analogické těm, které se nachází v knihovně `string.h` v ANSI C a nabízí řadu operací pro manipulaci a analýzu řetězců:)

`string strstr(str, index, len)`

Extrahuje podřetězec z `str`, začínající na indexu `idx` a pokrývající `len` znaků.

`long strlen(str)`

Vrací délku řetězce `str`, měřeno v znacích.

`long strfind(str, substr[, offset])`

Najde první výskyt `substr` v `str` a vrací jeho pozici. Prohledávání začíná od znaku s indexem `offset` (pokud není zadán, tak od začátku). Parametr `substr` může být i znak.

`long strrfind(str, substr)`

Identifikuje poslední výskyt `substr` v `str` a poskytuje jeho index.

strreplace(str, pattern, substr)
Nahrazuje všechny instance **pattern** v **str** řetězcem **substr**. Tato úprava je provedena přímo v **str**.

strupr(str)
Převéde všechny znaky v **str** na velká písmena.

strlwr(str)
Převéde **str** na malá písmena.

strtrim(str)
Odstraní počáteční a koncové bílé znaky z **str**.

long str2long(str[, default])
Převéde **str** na celé číslo. Pokud konverze selže, je vrácen volitelný druhý parametr (výchozí hodnota) nebo 0, pokud není poskytnut.

double str2double(str[, default])
Převéde **str** na desetinné číslo. Podobně jako **str2long**, vrací volitelnou výchozí hodnotu nebo 0 v případě selhání.

string long2str(num[, radix])
Převéde celé číslo **num** na řetězec s volitelným parametrem **radix**, který určuje základ (výchozí je 10). Výstupní řetězec neudává číselný systém (bez prefixů jako 0x pro hexadecimální).

string double2str(num[, precision])
Převéde desetinné číslo **num** na jeho řetězcovou reprezentaci s volitelným parametrem **precision**, který určuje maximální počet platných číslic.

strcpy(dest, src)
Zkopíruje obsah **src** do **dest**. Pro kompatibilitu s ANSI C dosahuje stejného výsledku **dest = src**.

strcat(dest, src)
Připojí **src** na konec **dest**. Stejně jako v ANSI C, **dest = dest + src** provádí stejnou operaci.

strcmp(str1, str2)
Porovná dva řetězce **str1** a **str2**. Pro stejný účel lze použít konstrukci **str1 == str2**, což je kompatibilní s ANSI C.

float2buf(buf, x[,endian])
Převéde reálné číslo **x** do pole **buf** čtyř prvků, z nichž každý reprezentuje jeden byte čísla uloženého ve formátu IEEE 754 single precision (známé jako float). Funkce je užitečná pro naplnění komunikačních bufferů. Volitelný 3. parametr má následující význam:
0 nativní endian procesoru (výchozí),
1 little endian,
2 big endian.

double2buf(buf, x[,endian])
Podobně jako **float2buf**, ale pro dvojitou přesnost, ukládá osm prvků (typ double).

double buf2float(buf[, endian])
Inverzní funkce k **float2buf**.

`double buf2double(buf[, endian])`

Inverzní funkce k `double2buf`.

`long RegExp(str, regexp, capture[])`

Porovnává `str` s regulárním výrazem `regexp`, ukládá zachycené skupiny do `capture`. Vrací počet zachycených skupin nebo záporný kód chyby.

Syntaxe regulárního výrazu zahrnuje standardní konstrukce jako:

- (?i) Musí být na začátku regulárního výrazu. Umožňuje porovnání bez ohledu na velikost písmen.
- ^ Porovnává začátek řetězce
- \$ Porovnává konec řetězce
- () Seskupování a zachycování podřetězců
- \s Porovnává bílý znak
- \S Porovnává ne-bílý znak
- \d Porovnává desetinnou číslici
- \n Porovnává znak nového řádku
- \r Porovnává znak pro posun řádku
- \f Porovnává znak vertikální tabulace
- \v Porovnává znak horizontální tabulace
- \t Porovnává znak horizontální tabulace
- \b Porovnává znak zpětného posunu
- + Porovnává jeden nebo vícekrát (nenasytně)
- +? Porovnává jeden nebo vícekrát (nenenasytně)
- * Porovnává nula nebo vícekrát (nenasytně)
- *? Porovnává nula nebo vícekrát (nenenasytně)
- ? Porovnává nula nebo jednou (nenenasytně)
- x|y Porovnává x nebo y (operátor alternace)
- \meta Porovnává jeden z meta znaků: ^\$().[]*+?|\
- \xHH Porovnává bajt s hexadecimální hodnotou 0xHH, např. \x4a.
- [...] Porovnává jakýkoliv znak ze sady. Podporovány jsou rozsahy jako [a-z].
- [^...] Porovnává jakýkoliv znak kromě těch ze sady.

Příklad:

```
RegExp("48,string1,string2","^(\\d+),([^\,]+)","capture");
```

Výsledek: `capture=["48,string1","48","string1"]`

`long ParseJson(json, cnt, names[], values[])`

Tato funkce zpracovává `json` řetězec, extrahuje hodnoty specifikovaných objektů. Pole `names` by mělo obsahovat názvy požadovaných vlastností (přístup k podpoložkám pomocí `.` a indexů pole pomocí `[]`, například `"cars[1].model"`). Odpovídající hodnoty jsou poté umístěny do pole `values`. Parametr `cnt` určuje počet objektů k analýze, který by měl odpovídat délce obou polí `names` a `values`. Funkce vrací celkový počet úspěšně analyzovaných hodnot nebo zápornou hodnotu, pokud dojde během analýzy k chybě.

Poznámka: Řetězcové proměnné jsou deklarovány stejně jako v ANSI C

(char <název proměnné>[<max znaků>];). Pro argumenty funkcí použijte char <název proměnné>[] nebo string <název proměnné>.

- **Systémové funkce** (nejsou v ANSI C)

Archive(arc, type, id, lvl_cnt, value)

Tato funkce archivuje hodnotu do archivačního podsystému systému. **arc** slouží jako bitová maska pro určení cílových archivů (například pro archivy 3 a 5 nastavte **arc** = 20, což je binárně 10100 nebo 20 v desítkové soustavě). Číslování archivů začíná od 1, s maximem 15 archivů (archiv 0 je vyhrazen pro interní systémové logy). Parametr **type** definuje datový typ s možnostmi:

- 1 Bool
- 2 Byte (U8)
- 3 Short (I16)
- 4 Long (I32)
- 5 Word (U16)
- 6 DWord (U32)
- 7 Float (F32)
- 8 Double (F64)
- 9 Time
- 10 Large (I64)
- 11 Error
- 12 String
- 17 Bool Group
- 18 Byte Group (U8)
- 19 Short Group (I16)
- 20 Long Group (I32)
- 21 Word Group (U16)
- 22 DWord Group (U32)
- 23 Float Group (F32)
- 24 Double Group (F64)
- 25 Time Group
- 26 Large Group (I64)
- 27 Error Group

id představuje unikátní ID archivní položky, **lvl_cnt** označuje úroveň plochy nebo počet prvků pro skupinové typy a **value** jsou data k archivaci.

Trace(id, val)

Zobrazuje hodnoty **id** a **val**. Funkce je určena zejména pro účely ladění. **id** je uživatelem definovaná konstanta v rozsahu od 0 do 9999 pro snadnou identifikaci zprávy. **val** může být jakýkoliv datový typ, včetně řetězců. Výstup se objevuje v systémovém logu REXYGEN.. Pro zobrazení těchto ladících zpráv v systémovém logu je nutné je povolit. Jděte do menu *Target→Diagnostic messages* a zaškrtněte políčko *Information* v boxu *Function block messages*. Logování musí být také povoleno pro daný blok

zaškrtnutím políčka *Enable logging* v záložce *Runtime* v dialogu parametrů bloku. Ve výchozím stavu je toto povoleno po umístění nového bloku z knihovny. Teprve poté jsou zprávy zobrazeny v *Systémovém logu*.

TraceError(id, val) TraceWarning(id, val) TraceVerbose(id, val)

Tyto příkazy mají podobný význam jako příkaz **Trace**, avšak výpis se objeví v jiné skupině systémového logu (*Error*, *Warning*, *Verbose*). Výpisy úrovně *Error* se do logu zapisují vždy, nezávisle na zaškrtnutí checkboxu *Enable logging* daného bloku. Zprávy úrovně *Warning* a *Verbose* je potřeba nejprve povolit, stejně jako v případě příkazu **Trace**.

Suspend(sec)

Přeruší provádění kódu skriptu, pokud od jeho puštění (v dané periodě) uplynulo více času (v sekundách), než je uvedeno. Při dalším spuštění bloku se pokračuje za tímto příkazem. Při **Suspend(0)** dojde k přerušení vždy.

double GetPeriod()

Vrací vlastní periodu spouštění daného bloku ve vteřinách.

double CurrentTime()

Vrací aktuální čas (v interním formátu). Používá se ve spojení s funkcí **ElapsedTime()**.

double ElapsedTime(new_time, old_time)

Vrací uplynulý čas v sekundách (desetinné číslo), tj. rozdíl mezi časy určenými parametry *new_time* a *old_time*, získaným z předchozího volání funkce **CurrentTime()**.

double Random()

Vrací pseudonáhodné číslo z intervalu $(0,1)$. Před voláním funkce **init()** se automaticky inicializuje generátor pseudonáhodného čísla, takže sekvence je vždy stejná.

long QGet(var)

Vrací kvalitu proměnné *var* (tak jak s ní pracuje systém REXYGEN, viz bloky **QFC**, **QFD**, **VIN**, **VOUT**). Funkci je možno použít jen pro vstupy, parametry a výstupy - pro vnitřní proměnné vrací vždy 0.

void QSet(var, value)

Nastaví kvalitu proměnné *var* (tak jak s ní pracuje systém REXYGEN) na hodnotu *val*. Funkci je možno použít jen pro výstupy - pro ostatní se nic nenastaví.

long QPropag([n,]val1, ..., valn)

Vrací kvalitu, která vznikne sloučením kvalit *val1, ..., valn*. Základní pravidlo pro slučování je, že výsledná kvalita je nejhorší ze vstupních. Pokud nastavíme kvalitu výstupu bloku s použitím této funkce, tak že do parametrů dáme kvalitu všech vstupů bloku, které výstup ovlivňují, dostaneme stejné chování jako u ostatních bloků systému REXYGEN.

double LoadValue(fileid, idx)

Čte hodnotu ze souboru. Podporuje jak binární soubory (s hodnotami **double**) tak textové soubory (hodnoty na samostatných řádcích). Soubor

je identifikován `fileid`, a `idx` udává index (pro binární soubory) nebo číslo řádku (pro textové soubory). V současnosti jsou podporovány následující hodnoty:

- 0 soubor na disku identifikovaný parametrem `p0`,
- 1 soubor na disku identifikovaný názvem bloku `REXLANG` a příponou `.dat`,
- 2 soubor na disku identifikovaný parametrem `srcname`, ale přípona je změněna na `.dat`,
- 3 soubor `rexlang.dat` v aktuálním adresáři,
- 4-7 stejné jako 0-3, ale formát je textový soubor. Každý řádek obsahuje jedno číslo. Index `idx` je číslo řádku a začíná na nule. Hodnota `idx=-1` znamená další řádek (např. sekvenční zápis).

`void SaveValue(fileid, idx, value)`

Ukládá hodnotu do souboru. Parametry jsou podobné jako u `LoadValue`.

`void GetSystemTime(time)`

Vrací systémový čas v UTC (upravitelný prostřednictvím nastavení operačního systému). Parametr `time` musí být pole alespoň 8 položek typu `long`. Funkce vyplní pole následujícími hodnotami v daném pořadí: rok, měsíc, den (v měsíci), den v týdnu, hodiny, minuty, sekundy, milisekundy. Na některých platformách má hodnota milisekund omezenou přesnost nebo není vůbec k dispozici (funkce vrací 0 ms).

`void Sleep(seconds)`

Zastaví provádění algoritmu bloku (a celého tasku) na definovanou dobu. Používejte tento blok s extrémní opatrností a pouze v případě, že neexistuje jiná možnost, jak dosáhnout požadovaného chování algoritmu. Doba usnutí by neměla překročit 900 milisekund. Nejkratší interval je asi 0,01s, přesná hodnota závisí na cílové platformě.

`long GetExtInt(ItemID)`

Vrací hodnotu vstupu/výstupu/parametru libovolného bloku v algoritmu `REXYGEN`. Tato externí datová položka je odkazována parametrem `ItemID`. Struktura řetězcového parametru `ItemID` je stejná jako například v parametru `sc` bloku `GETPI`. Pokud nelze hodnotu získat (např. neplatné nebo neexistující `ItemID`, konflikt datového typu atd.), blok `REXLANG` ohlásí chybu a musí být resetován.

`long GetExtLong(ItemID)`

Viz `GetExtInt(ItemID)`.

`double GetExtReal(ItemID)`

Stejný význam jako předchozí příkazy, ale pro desetinné číslo.

`double GetExtDouble(ItemID)`

Viz `GetExtReal(ItemID)`.

`string GetExtString(ItemID)`

Stejný význam jako předchozí příkazy, ale pro řetězec/text.

void SetExt(ItemID, value)

Nastaví hodnotu vstupu/výstupu/parametru libovolného bloku v REXu určeného parametrem `ItemID`. Tento parametr je textový a má stejný význam/strukturu jako parametr `sc` bloku `SETPI`. Nastavuje se hodnota parametru `value`, přičemž typ nastavené hodnoty (long/double/string) je určen typem parametru `value`. Pokud hodnotu nelze nastavit (např. neexistující `ItemID` nebo neodpovídá datový typ) blok `REXLANG` skončí chybou a je potřeba ho resetovat.

int BrowseExt(ItemID, first_subitem_index, max_count, subitems, kinds)

Funkce prochází adresový prostor úlohy. Pokud je `ItemID` identifikátorem bloku (cestou k bloku), pole řetězců `subitems` bude obsahovat názvy všech vstupů, výstupů, parametrů a vnitřních stavů. Funkce vrátí počet podpoložek nebo záporný chybový kód. Hodnoty `kinds` jsou: executive = 0, module = 1, driver = 2, archive = 3, level = 4, task = 5, quicktask = 6, subsystem = 7, block = 8, input = 9, output = 10, internal state = 11, parameter or state array = 12, special = 13.

long CallExt(ItemID)

Spustí jeden krok jakéhokoli bloku v algoritmu REXYGEN, identifikovaného pomocí `ItemID`. Struktura textového parametru `ItemID` je stejná jako například u parametru `sc` bloku `GETPI`. Doporučuje se volat pouze pozastavené bloky², které by měly být ve stejném úkolu jako blok `REXLANG`. Funkce vrátí výsledkový kód volaného bloku (viz chybové kódy REXYGEN).

long GetInArrRows(input)

Vrací počet řádek pole připojeného ke vstupu bloku `REXLANG` s indexem `input`.

long GetInArrCols(input)

Vrací počet sloupců pole připojeného ke vstupu bloku `REXLANG` s indexem `input`.

long GetInArrMax(input)

Vrací maximální (alokovanou) velikost pole připojeného ke vstupu bloku `REXLANG` s indexem `input`.

double GetInArrDouble(input, row, col)

Vrací prvek pole připojeného ke vstupu bloku `REXLANG` s indexem `input`.

long GetInArrLong(input, row, col)

Stejně jako `GetInArrDouble(...)`, ale hodnota je typu long.

Void SetInArrValue(input, row, col, value)

Nastaví prvek pole připojeného ke vstupu bloku `REXLANG` s indexem `input`.

Void SetInArrDim(input, row, col)

Nastaví rozměr pole připojeného ke vstupu bloku `REXLANG` s indexem `input`.

²nastavte zaškrťovací políčko `Halt` na stránce vlastností `Runtime` v dialogu nastavení parametrů bloku

```
long memrd32(hMem, offset)
    Čtení fyzické paměti. Handle se získá pomocí funkce
    Open(72, "/dev/mem", <fyzická adresa>, <velikost oblasti>).
long memwr32(hMem, offset, value)
    Zápis do fyzické paměti. Handle se získá funkce
    OpenMemory("/dev/mem", <physical address>, <area size>).
```

- **Komunikační funkce** (nejsou v ANSI C)

Tento soubor funkcí usnadňuje komunikaci přes různé kanály včetně TCP/IP, UDP/IP, sériových linek (RS-232 nebo RS-485), SPI sběrnice a I2C sběrnice. Níže je stručný seznam těchto funkcí. Pro podrobné informace viz příkladové projekty v systému REXYGEN.

```
long OpenFile(string filename)
    Tato funkce otevře soubor a vrátí identifikační číslo (handle) souboru.
    Pokud funkce vrátí zápornou hodnotu, otevření souboru nebylo úspěšné.
long OpenCom(string comname, long baudrate, long parity)
    Otevírá sériovou linku a vrátí její handle. Od verze 3.0 REXYGEN lze jako
    comname specifikovat virtuální porty. Pro podrobné informace o virtuálních
    portech viz popis bloku UART. Pokud je vrácena záporná hodnota, otevření
    selhalo. Možnosti nastavení parity jsou 0 pro žádnou, 1 pro lichou a 2 pro
    sudou.
long OpenUDP(string localname, long lclPort, string remotename, long remPort)
    Otevírá UDP socket a vrátí jeho handle. Pokud funkce vrátí zápornou
    hodnotu, otevření socketu nebylo úspěšné. Funkce může otevřít buď IPv4
    nebo IPv6 socket na základě remotename, localname nebo nastavení ope-
    račního systému, pokud je použito DNS jméno. Volitelná nastavení zahr-
    nují prázdný localname (jakékoli rozhraní), prázdný remotename nebo 0
    pro remPort (nepoužívá se) a 0 pro lclPort (přiděleno UDP/IP stackem).
long OpenTCPSvr(string localname, long lclPort)
    Tato funkce otevírá TCP socket v režimu serveru (poslechu). Vrací handle
    socketu a záporná návratová hodnota znamená neúspěšné otevření. Funkce
    může otevřít buď IPv4 nebo IPv6 socket v závislosti na remotename,
    localname nebo nastavení operačního systému, pokud je použito DNS
    jméno. Můžete nastavit prázdný localname jako jakékoli rozhraní.
long OpenTCPcli(string remotename, long remPort)
    Otevírá TCP socket v režimu klienta a vrátí jeho handle. Záporná návra-
    tová hodnota znamená selhání otevření socketu. Funkce otevírá buď IPv4
    nebo IPv6 socket na základě remotename, localname nebo nastavení ope-
    račního systému, pokud je použito DNS jméno. Všimněte si, že tato funkce
    nečeká na navázání spojení, což může trvat několik milisekund v místní
    síti nebo několik sekund pro vzdálené lokace. Pokud jsou volány Write()
    nebo Read() před navázáním spojení, je vrácen chybový kód -307 (chyba
    otevření souboru).
```

`long OpenI2C(string devicename)`

Otevírá I2C sběrnici a vrátí její handle. Pokud funkce vrátí zápornou hodnotu, otevření nebylo úspěšné.

`long OpenSPI(string devicename)`

Otevírá SPI sběrnici a vrátí její handle. Záporná návratová hodnota znamená neúspěšné otevření.

`long OpenDevice(string filename)`

Podobně jako `OpenFile()`, ale funkce `Write()` a `Read()` jsou neblokující. Pokud nelze data číst nebo zapisovat, funkce okamžitě vrátí chybový kód -1.

`long OpenMemory(string devicename, long baseaddr, long size)`

Mapuje fyzickou paměť a vrátí přidružený handle. Pokud je vrácena záporná hodnota, operace nebyla úspěšná.

`long OpenSHM(string devicename, long deviceid, long size, long flags)`

Mapuje sdílenou paměť (pouze v Linuxu, využívá `ftok()` a `shmget()`) a vrací přidružený handle. První a druhý parametr slouží k identifikaci paměťové oblasti (musí být shodné pro všechny kooperující entity). Parametr `size` udává velikost sdílené paměťové oblasti v bajtech. Parametr `flags` reprezentuje standardní linuxové příznaky a oprávnění (pokud je nastaveno na 0, což je výchozí hodnota, jsou nastavena následující práva: vytvořit oblast, pokud neexistuje, a umožnit všem čtení i zápis).

`void Close(long handle)`

Zavře socket, sériovou linku, soubor nebo jakékoli zařízení otevřené pomocí funkce `Open...`

`void SetOptions(long handle, long params[])`

Nastavuje parametry socketu nebo sériové linky. Velikost pole musí být alespoň:

- 22 pro sériovou linku (na Windows parametry pro SetCommState() a SetCommTimeouts() v následujícím pořadí: BaudRate, fParity, Parity, StopBits, ByteSize, fDtrControl, fRtsControl, fAbortOnError, fBinary, fErrorChar, fNull, fDsrSensitivity, fInX, fOutX, fOutxCtsFlow, fOutxDsrFlow, fTXContinueOnXoff, ReadIntervalTimeout, ReadTotalTimeoutConstant, ReadTotalTimeoutMultiplier, WriteTotalTimeoutConstant, WriteTotalTimeoutMultiplier; linux používá odlišnou funkci, ale význam parametrů je stejný, pokud je to možné),
 - 2 pro soubor (1. položka je mód: 1=vyhledání od začátku, 2=vyhledání od aktuální pozice, 3=vyhledání od konce, 4=nastavit konec souboru, 2. položka je posun pro vyhledání),
 - 3 pro SPI (1. položka je režim SPI, 2. položka je počet bitů na slovo, 3. položka je maximální rychlost v Hz),
 - 5 pro I2C (1. položka je adresa zařízení, 2. položka je příznak 10bitové adresy, 3. položka je příznak kontroly chyb v balíčku, 4. položka je počet pokusů, 5. položka je časový limit),
- ostatní typy handle nejsou podporovány

`void GetOptions(long handle, long params[])`

Načte parametry socketu nebo sériové linky a uloží je do pole `params`. Velikost pole musí odpovídat požadavkům daného zařízení (viz `SetOptions`).

`long Accept(long hListen)`

Přijme spojení na poslouchajícím socketu a vrátí handle komunikačního socketu nebo chybový kód.

`long Read(long handle, long buffer[], long count[, offset])`

Přijímá data ze sériové linky nebo socketu a vrátí počet přečtených bajtů nebo chybový kód. Parametr `count` definuje maximální počet bajtů k přečtení. Každý bajt přichozících dat je vložen do pole `buffer` typu `long` ve stejném pořadí. Funkce má ještě jeden (nepovinný) parametr `offset`, který lze použít při čtení dat z paměti, kdy je handle vytvořen pomocí `OpenSHM()` nebo `OpenMemory()`.

Lze také použít zápis

`long Read(long handle, string data[], long count)` (tj. používá se řetězec místo pole dat; jeden bajt ve vstupním souboru odpovídá jednomu znaku; nelze použít pro binární soubory).

Chybové kódy jsou:

- 1 je nutné počkat na dokončení operace (funkce je "nebloku-
jící")
- 309 čtení selhalo; chybový kód operačního systému se zobrazí v
logu (pokud je povoleno logování bloků funkcí)
- 307 soubor/socket není otevřen

long Write(long handle, long buffer[], long count[, offset])

Odesílá data přes sériovou linku nebo socket. Parametr **count** definuje počet bajtů k odeslání. Počet odeslaných bajtů nebo chybový kód je vrácen. Každý bajt odchozích dat je přečten z pole **buffer** typu **long** ve stejném pořadí. Funkce má ještě jeden (nepovinný) parametr **offset**, který lze použít pro zápis dat do paměti, kdy je handle vytvořen pomocí **OpenSHM()** nebo **OpenMemory()**.

Lze také použít formu

long Write(long handle, string data) (tj. používá se řetězec místo pole dat; jeden bajt ve výstupním souboru odpovídá jednomu znaku; nelze použít pro binární soubory).

Chybové kódy jsou:

- 1 je nutné počkat na dokončení operace (funkce je "nebloku-
jící")
- 310 zápis selhal; chybový kód operačního systému se zobrazí v
logu (pokud je povoleno logování bloků funkcí)
- 307 soubor/socket není otevřen

long ReadLine(long handle, string data)

Čte řádek z (textového) souboru, sériové linky nebo socketu a ukládá znaky do **data** až do jejich alokované velikosti. Funkce vrátí skutečnou velikost řádku nebo chybový kód.

long DeleteFile(string filename)

Smaže soubor. Vrací 0 pokud je soubor smazán; záporné číslo znamená chybu.

long RenameFile(string filename, string newfilename)

Přejmenuje soubor. Vrací 0 pokud je soubor přejmenován; záporné číslo znamená chybu.

bool ExistFile(string filename)

Ověří, zda soubor nebo zařízení existuje (lze otevřít pro čtení), vrátí true nebo false.

long I2C(long handle, long addr, long bufW[], long cntW, long bufR[], long cntR)

Zajišťuje komunikaci přes sběrnici I2C, zejména na Linuxových systémech s možností I2C (např. Raspberry Pi). Funkce odesílá a přijímá data do/z slave zařízení pomocí **addr**. Parametr **handle** je vrácen funkcí **OpenI2C**, jejíž parametr určuje název zařízení (podle operačního systému). Parametr **bufW** je buffer (pole) pro data, která se odesílají, **cntW** je počet bajtů k odeslání, **bufR** je buffer (pole) pro příchozí data a **cntR** je počet bajtů k přijetí. Funkce vrací 0 nebo chybový kód.

long SPI(long handle, 0, long bufW[], long cntW, long bufR[], long cntR)

Provede transakci přes sběrnici SPI, zejména na Linuxových systémech se sběrnici SPI. Parametr **handle** je vrácen funkcí **OpenSPI**, jejíž parametr určuje název zařízení (podle operačního systému). Druhý parametr je vždy 0 (rezervováno pro interní použití). Parametr **bufW** je buffer (pole) pro data, která se odesílají, **cntW** je počet bajtů k odeslání, **bufR** je buffer

(pole) pro příchozí data a `cntR` je počet bajtů k přijetí. Všimněte si, že komunikace přes sběrnici SPI je full-duplex, proto výsledná délka transakce SPI je dána maximem parametrů `cntW` a `cntR`, nikoli jejich součtem. Funkce vrací 0 nebo chybový kód.

`long Seek(long handle, long mode[], long offset)`

Nastaví pozici pro příkazy Read/Write. Parametr `mode` znamená:

- 1 posun od začátku souboru,
- 2 posun od aktuální pozice,
- 3 posun od konce souboru.

`long Recv(long handle, long buffer[], long count)`

Zastaralá funkce. Použijte Read.

`long Send(long handle, long buffer[], long count)`

Zastaralá funkce. Použijte Write.

`long crc16(data, length, init, poly, flags, offset)`

Vypočítá 16bitový cyklický redundantní kód (CRC), běžně používaný jako kontrolní součet nebo hash v různých komunikačních protokolech.

data Pole bajtů (reprezentováno polem typu `long`) nebo řetězec, pro který se vypočítává hash.

length Počet bajtů ve vstupním poli nebo textu. Použijte -1 pro zpracování celého řetězce.

init Počáteční vektor pro výpočet CRC.

poly Kontrolní polynom použitý ve výpočtu.

flags Konfigurační příznaky:

- 1: Obrací pořadí bitů jak ve vstupních bajtech, tak ve výsledném CRC.
- 2: Výsledné CRC je XORováno s 0xFFFF.
- 4: Pokud je **data** pole typu `long`, jsou zpracovány všechny 4 bajty v prvku pole (LSB první).
- 8: Podobně jako příznak 4, ale zpracovává MSB první.

offset Index prvního zpracovávaného bajtu v datovém poli (obvykle 0).

Poznámka: Existují analogické funkce pro výpočet 32bitového a 8bitového CRC: `long crc32(data, length, init, poly, flags, offset)` a `long crc8(data, length, init, poly, flags, offset)`. Počáteční vektor, kontrolní polynom a příznaky pro různé protokoly lze nalézt na <https://crccalc.com/>

Příklady:

- MODBUS: `crc16("123456789", -1, 0xFFFF, 0x8005, 1, 0)`
- DECT-X: `crc16("123456789", -1, 0, 0x0589, 0, 0)`

Další poznámka: Funkce `crc8(...)` a `crc32(...)` také existují, podporují výpočty 8bitových a 32bitových CRC se stejnou strukturou parametrů.

Ladění kódu, debugging

Pro ladění kódu je k dispozici příkaz **Trace**, viz výše. Dále lze použít výstupy bloku, které se nepoužívají pro vlastní algoritmus a zapisovat do nich hodnoty různých mezivýpočtů. V závislosti na povaze algoritmu může být vhodné tyto ladící hodnoty připojit do trendu. Pokud je potřeba sledovat hodnot více, je možné do tasku přidat blok CNA (připojený na TRNDV nebo VTOR) a do hodnot v jeho poli nastavovat opět různé mezivýsledky pomocí funkce **SetExt**.

Poznámky

- Datové typy vstupů **u0..u15**, výstupů **y0..y15** a parametrů **p0..p15** jsou určeny během kompilace zdrojového kódu.
- Chybové kódy **< -99** vyžadují vstup **RESET** pro restartování bloku **REXLANG** po řešení příčiny chyby.
- **POZOR!!!** Ve funkci **init()** je sice možné číst vstupy, ale protože ostatní bloky obvykle nenastavují v **init** fázi výstupy, bude tam vždy 0. Nastavovat výstupy lze, ale obvykle se to nedělá.
- Skript může obsahovat cestu v parametru **srcname**; pokud ne, očekává se soubor v adresáři projektu nebo ve specifikovaných adresářích (viz parametr **LibraryPath** bloku [PARAM](#)).
- Parametr **srcname** je možné udávat s absolutní cestou. V opačném případě se soubor hledá na aktuálním adresáři a specifikovaných adresářích (viz parametr **LibraryPath** bloku [PARAM](#)).
- Parametry vektorových funkcí jsou primárně typu **double**, s výjimkou parametru **n**, který je typu **long**. Funkce s jedním vektorovým parametrem existují ve třech variantách:

```
double function(val1,...,valn)
```

Vektor je definován jako sekvence hodnot typu **double**.

```
double function(n,val1,...,valn)
```

Vektor je definován jako v prvním případě, pouze první parametr definuje počet hodnot – velikost vektoru. Tato varianta je kompatibilní s kompilátorem jazyka C. Parametr **n**³ musí být číslo, nikoli tzv. **const** proměnná, a musí odpovídat počtu následujících prvků definujících vektor.

```
double function(n,vec)
```

Parametr **n** je libovolný výraz typu **long** a definuje počet prvků, které funkce zohledňuje.

³Volitelný parametr **n** vektorových funkcí musí být specifikován, pokud je vyžadována kompatibilita s kompilátorem jazyka C/C++. V takovém případě musí být implementovány také všechny nestandardní funkce a funkce s proměnným počtem parametrů musí znát počet parametrů.

- Je důležité si pamatovat, že pole v skriptovacím jazyku se chovají podobně jako pole v jazyce C: indexace začíná od 0 a neexistuje automatická kontrola hranic. Například pokud deklarujete `double vec[10], x;`, pole `vec` bude mít prvky indexované od 0 do 9. Přístup k `vec[10]` nevyvolá chybu syntaxe ani runtime, ale vrácená hodnota je nedefinovaná, protože leží mimo hranice pole. Navíc přiřazení hodnoty k `vec[11]` (např. `vec[11] = x;`) může být obzvlášť nebezpečné, protože může nechtěně přepsat sousední paměťová místa. To by mohlo vést k nepředvídatelnému chování nebo způsobit pád programu.
- Během kompilačního procesu, pokud existují chyby syntaxe, kompilátor hlásí **parser error** spolu s číslem řádku, kde došlo k chybě. Tyto hlášení konkrétně ukazují na problémy se syntaxí. Pokud však syntaxe vypadá správně a chyba je stále hlášena, je vhodné zkontrolovat konflikty zahrnující identifikátory, klíčová slova nebo názvy funkcí, jelikož tyto mohou také způsobit chyby neokamžitě zřejmé jako chyby syntaxe.
- Všechny skoky jsou přeloženy jako relativní, tj. odpovídající kód je omezen na 32767 instrukcí (ve formátu přenosném pro různé platformy).
- Všechny platné proměnné a výsledky dočasných výpočtů jsou uloženy v zásobníku, který zahrnuje:
 - Globální a lokální **static** proměnné jsou trvale umístěny na začátku zásobníku.
 - Návrátové adresy pro volání funkcí.
 - Parametry funkcí.
 - Proměnné lokální pro funkce.
 - Návrátová hodnota funkce.
 - Dočasné výsledky výpočtů. Například ve výrazu `a = b + c;` jsou nejprve `b` a `c` vloženy na zásobník. Jejich součet je pak vypočítán, operandy jsou odstraněny ze zásobníku a výsledek je vložen na zásobník.

Jednoduché proměnné, jako jsou **long** nebo **double**, zabírají jeden slot zásobníku. U polí záleží na celkové velikosti, bez ohledu na typ prvků.

- Když jsou pole předávána funkcím, jsou referencována, nikoli kopírována. To znamená, že pro referenci je použit pouze jeden slot zásobníku a funkce přímo pracuje s původním polem.
- Pokud je alokovaná velikost zásobníku nedostatečná (méně než prostor potřebný pro globální proměnné plus 10), automaticky se zdvojnásobí, s dalšími 100 sloty pro výpočetní potřeby, parametry funkcí a lokální proměnné, zejména pokud je definováno málo globálních proměnných.

- Při základní úrovni ladění jsou během provádění skriptu prováděny různé kontroly. Ty zahrnují inicializaci čtených hodnot a kontrolu hranic indexů polí. Navíc je na začátek a konec každého deklarovaného pole vloženo několik neinicializovaných hodnot pro kontrolu hranic a do souboru *.ill jsou přidány instrukce NOP s čísly řádků zdrojového souboru.
- Při úplné úrovni ladění je povolena další kontrola neplatných přístupů k datovým rozsahům (jako jsou přetečení zásobníku).
- V tomto kontextu se 'instrukcí' rozumí procesorově nezávislý mnemonický kód. Tyto kódy jsou uloženy v souboru *.ill.
- Funkce `OpenCom()` nastaví binární neblokující režim bez timeoutů, 8 datových bitů, 1 stopbit, bez parity, 19200Bd. Volitelně lze upravit `baudrate` a `parity` parametr ve funkci `OpenCom()`.
- Přístup k textovým souborům je výrazně pomalejší než k binárním souborům. Textové soubory však mají výhodu, že jsou viditelné a upravitelné bez specializovaného software.
- Blok automaticky nevyvolá funkci `parchange()`. Tato funkce musí být manuálně volána v rámci funkce `init()`, pokud je to potřeba.
- Funkce `OpenFile()` otevírá soubory v datovém adresáři systému REXYGEN (tj. v Linuxu implicitně v `\rex\data`, na Windows `C:\ProgramData\REX Controls\REX_<verze>\RexCore`). Jsou dovoleny podadresáře, ale není dovoleno `..\`. Linky se následují.

Ladění kódu

Použijte výše uvedený příkaz `Trace`.

Příklady inicializace polí

Níže je uvedeno několik příkladů inicializace polí v C-like jazyku. Pole je možné použít ve vektorových funkcích.

```
double vec0[20]; // Neinicializované pole/vektor. Globální proměnné vč.
                // polí jsou automaticky inicializované na 0.
double vec1[20] = {0, 1.1, 2.2, 3, 4}; // Částečně inicializované pole/
                // vektor.
double vec2[] = {10, 11.1, 12.2, sqrt(2), 14}; // Inicializované pole s
                // automaticky určenou velikostí (5 prvků).

char str0[30]; // Neinicializovaný textový řetězec s kapacitou 29 znaků
                // plus ukončovací nulový znak.
char str1[30] = "ahoj"; // Inicializovaný textový řetězec.

int mat0[5][10]; // Neinicializovaná matice s 5 řádky a 10 sloupci.
int mat1[5][6] = {{100, 101, 102, 103}, {110, 111, 112, 113}, {120, 121,
                122, 123, 124}}; // Částečná inicializace matice.
```

```
int mat2[5][6] = {{200, 201, 102}, {210, 211, 212, 213}, {220, 221, 222,
223, 224, 225}, {230}, {240,241}};

char strV0[4][30]; // Neinicializovaný vektor čtyř textových řetězců.
char strV1[4][30] = {"jméno", "příjmení", "adresa"}; // Částečně
inicializovaný vektor textů.
```

Příklad C-like

Následující příklad ukazuje jednoduchý kód pro sečtení dvou vstupních signálů a také sečtení dvou uživatelsky definovaných parametrů.

```
double input(0) input_u0;
double input(2) input_u2;

double parameter(0) param_p0;
double parameter(1) param_p1;

double output(0) output_y0;
double output(1) output_y1;

double my_value;

long init(void)
{
    my_value = 3.14;
    return 0;
}

long main(void)
{
    output_y0 = input_u0 + input_u2;
    output_y1 = param_p0 + param_p1 + my_value;
    return 0;
}

long exit(void)
{
    return 0;
}
```

Příklad STL

A zde je stejný příklad ve strukturovaném textu.

```
VAR_INPUT
    input_u0:REAL;
    input_u1:REAL;
    input_u2:REAL;
END_VAR

VAR_OUTPUT
    output_y0:REAL;
    output_y1:REAL;
```

```

END_VAR

VAR_PARAMETER
    param_p0:REAL;
    param_p1:REAL;
END_VAR

VAR
    my_value: REAL;
END_VAR

FUNCTION init : INT;
    my_value := 3.14;
    init := 0;
END_FUNCTION

FUNCTION main : INT;
    output_y0 := input_u0 + input_u2;
    output_y1 := param_p0 + param_p1 + my_value;
    main := 0;
END_FUNCTION

FUNCTION exit : INT;
    exit := 0;
END_FUNCTION

```

Tento blok nepropaguje kvalitu signálu. Více informací je uvedeno v sekci 1.4.

Vstup

HLD	Pozastavení	Bool
RESET	Reset bloku	Bool
u0..u15	Vstupní signál	Any

Parametr

srcname	Jméno souboru se skriptem	⊙srcfile.c	String
srctype	Typ zdrojového souboru	⊙1	Long (I32)
	1 C		
	2 STL		
	3 RLB		
	4 ILL		
debug	Úroveň/množství ladicích kontrol a informací	⊙3	Long (I32)
	1 bez kontroly		
	2 základní kontrola		
	3 úplná kontrola		
stack	Velikost zásobníku v bytech (0=automaticky)		Long (I32)
strs	Velikost bufferu pro řetězce (počet znaků, 0=automaticky)		Long (I32)
p0..p15	Parametr		Any

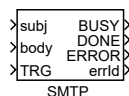
Výstup

iE	Kód chyby	Error
	i kód chyby REXYGENu	
y0..y15	Výstupní signál	Any

SMTP – Blok pro odesílání e-mailových oznámení přes SMTP

Symbol bloku

Licence: [ADVANCED](#)



Popis funkce

Blok **SMTP** odešle e-mailovou zprávu pomocí standardního protokolu SMTP. Blok funguje jako jednoduchý e-mailový klient. Neimplementuje mailový server.

Obsah zprávy je definován vstupy **subj** a **body**. Parametry **from** a **to** specifikují odesílatele a příjemce zprávy. Zpráva je odeslána, při detekci náběžné hrany (**off**→**on**) na vstupu **TRG**. Poté je výstup **BUSY** nastaven na **on**, dokud není požadavek dokončen, což je signalizováno výstupem **DONE**. V případě chyby je nastaven výstup **ERROR**. Výstup **errId** nese poslední chybu identifikovanou systémovým kódem chyby **REXYGEN**. Parametr **domain** musí být vždy nastaven pro identifikaci cílového zařízení. Výchozí hodnota by měla fungovat ve většině případů. Může být více příjemců zprávy. V takovém případě musí být jednotlivé e-mailové adresy odděleny čárkou a nesmí být přítomen žádný mezerový znak.

Blok může být spuštěn v neblokovacím nebo blokovacím režimu, což je specifikováno parametrem **BLOCKING**:

- V *blokovacím režimu* je vykonání úlohy pozastaveno, dokud není odesílání e-mailu dokončeno. Tento režim je obvykle používán v úlohách s dlouhou dobou vykonání, $T_S \geq 10s$. Pokud e-mail není úspěšně odeslán do doby, než vyprší **timeout**, je indikována chyba a vykonání úlohy je obnoveno.
- V *neblokovacím režimu* blok **SMTP** vykoná pouze jednu operaci v každém vykonání bloku a vykonání úlohy není pozastaveno. Tento režim je obvykle používán v úlohách s krátkou dobou vykonání, $T_S \leq 0.1s$. V tomto režimu by měl být parametr **timeout** nastaven na alespoň $50 \cdot T_S$, kde T_S je doba vykonání v sekundách.

Je doporučeno spouštět blok **SMTP** v neblokovacím režimu. Je však nutné zmínit, že na různých operačních systémech některé operace nemusí být v neblokovacím režimu podporovány, takže buďte opatrní a nepoužívejte tento blok v rychlých úlohách (viz [QTASK](#)) nebo v úlohách s extrémně krátkou dobou vykonání (několik milisekund). Nejlepší podpora neblokovacího provozu je na operačním systému GNU/Linux.

Blok podporuje ověření uživatele pomocí standardní metody ověřování SMTP. Uživatelské jméno a heslo lze specifikovat parametry **user** a **password**. Blok také podporuje zabezpečené spojení. Metoda šifrování je vybrána parametrem **tls**. Je také možné nechat blok ověřit certifikát serveru nastavením parametru **VERIFY**. SSL certifikát serveru

nebo důvěryhodný certifikační orgán serveru musí být uložen v parametru `certificate` ve formátu PEM. Blok nepodporuje žádné uložení certifikátů.

Délka celé zprávy (předmět, tělo a hlavičky) je omezena na maximálně 1024 znaků.

Tento blok nepropaguje kvalitu signálu. Více informací je uvedeno v sekci 1.4.

Vstup

<code>subj</code>	Předmět e-mailu	String
<code>body</code>	Tělo e-mailu	String
<code>TRG</code>	Spuštění zvolené akce	Bool

Parametr

<code>server</code>	Adresa SMTP serveru	String
<code>to</code>	E-mail příjemce	String
<code>from</code>	E-mail odesílatele	String
<code>tls</code>	Metoda šifrování	⊙1 Long (I32)
	1 None	
	2 StartTLS	
	3 TLS	
<code>user</code>	Uživatelské jméno	String
<code>password</code>	Heslo	String
<code>domain</code>	domain	String
<code>auth</code>	Metoda autentizace	⊙1 Long (I32)
	1 Login	
	2 Plain	
<code>certificate</code>	Certifikát pro autentizaci	String
<code>VERIFY</code>	Povolení verifikace serveru (platnost certifikátu)	Bool
<code>timeout</code>	Povolená doba pro dokončení operace	Double (F64)
<code>BLOCKING</code>	Čekání na dokončení operace	Bool

Výstup

<code>BUSY</code>	Odesílání e-mailu	Bool
<code>DONE</code>	E-mail byl odeslán	Bool
<code>ERROR</code>	Příznak chyby	Bool
<code>errId</code>	Kód chyby	Error

STEAM – Přepočet vlastností páry

Symbol bloku

Licence: [STANDARD](#)



Popis funkce

Blok **STEAM** slouží k výpočtu různých termodynamických parametrů vody a páry, jako jsou entalpie, entropie, teplota a tlak sytosti, hustota, viskozita a další. Vstupy bloku jsou typicky teplota a tlak, ale výpočty jsou možné i v opačném směru, například určení teploty z entalpie. Výpočty se řídí mezinárodním standardem **IAPWS IF-97**, jehož podrobnosti lze najít na <http://www.iapws.org/relguide/IF97-Rev.pdf>.

Jednotky pro teplotu lze nastavit parametrem **tunit**, pro tlak parametrem **punit**. Jednotky energie jsou v kilojoulech [kJ], například entalpie je vyjádřena v kJ/kg, tepelná kapacita v kJ/kg/K, což odpovídá definicím v IF-97. Ostatní veličiny jsou uvedeny v SI jednotkách, jako je hustota v kg/m³.

Funkce bloku má název ve formátu **<výstupní veličina>_<1.vstupní veličina><2.vstupní veličina>**, kde veličiny zahrnují:

- **T** - Teplota
- **p** - Tlak
- **h** - Entalpie [kJ/kg]
- **v** - Měrný objem [m³/kg]
- **rho** - Hustota [kg/m³]
- **s** - Měrná entropie
- **u** - Měrná vnitřní energie [kJ/kg]
- **Cp** - Měrná tepelná kapacita při konstantním tlaku [kJ/kg/K]
- **Cv** - Měrná tepelná kapacita při konstantním objemu [kJ/kg/K]
- **w** - Rychlost zvuku [m/s]
- **my** - Viskozita
- **tc** - Tepelná vodivost
- **st** - Povrchové napětí

- **x** - Hmotnostní podíl páry
- **vx** - Objemový podíl páry

Výstupní veličina může obsahovat modifikátory:

- **sat** - Saturační hodnota, tj. pro stav na přechodu mezi vodou a párou
- **V** - Pára (plyn) pro stav na přechodu mezi vodou a párou
- **L** - Voda (kapalina) pro stav na přechodu mezi vodou a párou

Příklady

- **h_pT** - Výstupem je entalpie pro zadaný tlak (1. vstup) a teplotu (2. vstup); například pro tlak 100 kPa a teplotu 120 °C je entalpie 2716 kJ/kg.
- **Tsat_p** - Výstupem je teplota sytosti pro zadaný tlak (1. vstup); například při tlaku 100 kPa je teplota sytosti 100 °C.
- **hL_p** - Výstupem je entalpie vody na mezi sytosti pro zadaný tlak (1. vstup); například při 100 kPa (a teplotě 100 °C, aby byl stav na mezi sytosti) může médium obsahovat libovolný poměr vody a páry, funkce (výstup bloku) bude entalpie pro situaci, že médium je voda (bez páry), tj. 417 kJ/kg.

Tento blok nepropaguje kvalitu signálu. Více informací je uvedeno v sekci [1.4](#).

Vstup

u1	První analogový vstup bloku	Double (F64)
u2	Druhý analogový vstup bloku	Double (F64)

Parametr

func	Typ funkce	⊙1 Long (I32)
punit	Jednotka tlaku	⊙1 Long (I32)
	1 MPa	
	2 bar	
	3 kPa	
tunit	Jednotka teploty	⊙1 Long (I32)
	1 K	
	2 °C	

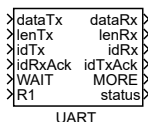
Výstup

y	Analogový výstupní signál	Double (F64)
E	Příznak chyby	Bool

UART – Komunikační blok pro UART

Symbol bloku

Licence: [STANDARD](#)



Popis funkce

Blok **UART** umožňuje číst a zapisovat data pomocí univerzálního asynchronního přijímače-vysílače. Parametr **port** specifikuje název zařízení. Je možné použít dva typy názvů:

- adresa fyzického zařízení – obvykle `/dev/ttyS*` pro cílový systém Linux nebo `COM*` pro Windows. Nahraďte symbol `"*"` podle zvoleného sériového portu!
- virtuální adresa – **REXYGEN** umožňuje vytvoření virtuálního UART, pomocí kterého můžete komunikovat uvnitř **REXYGEN** s jinými bloky podporujícími UART, jako jsou [REXLANG](#), [PYTHON](#), další blok **UART** nebo **Modbus driver**. Na zařízeních s operačním systémem Linux je virtuální port označen prefixem **pty:** (pseudo terminál) a je možné se k němu připojit z jiné aplikace běžící na zařízení. Na zařízeních s operačním systémem Windows je možné použít prefix **vcom**, který umožňuje komunikaci uvnitř **REXYGEN**. Příklady virtuálních portů: `pty:/tmp/vslave`, `vcom:vmaster`.

Komunikace pomocí UART má několik obecných vlastností, které jsou nastaveny pomocí parametrů, jako jsou **baudrate**, **parity**, **databits** a **stopbits**. Každému paketu, který je přijat nebo odeslán, je přiřazeno jedinečné ID. ID následujícího paketu je vždy o jedna vyšší než ID předchozího paketu. Jakmile je dosaženo maximální hodnoty ID, následující přiřazené ID bude 0. Maximální hodnota ID je určena parametrem **maxId**. Data jsou odesílána s náběžnou hranou vstupu **idTx**. Pokud se v bufferu stále nachází data k odeslání, je výstup **WAIT** nastaven na **on**.

Tento blok nepropaguje kvalitu signálu. Více informací je uvedeno v sekci [1.4](#).

Vstup

dataTx	Reference vektoru s vysílanými daty		Reference
lenTx	Délka vysílaných dat (0 = celý vektor)	↓0	Long (I32)
idTx	ID vysílaného packetu	↓0	Long (I32)
idRxAck	ID posledního zpracovaného přijatého packetu	↓0	Long (I32)
WAIT	Příznak pozastavení vysílání (data jsou vložena do mezibufferu)		Bool
R1	Reset bloku		Bool

dataRx	Reference vektoru s přijatými daty		Reference
lenRx	Délka přijatých dat	↓0	Long (I32)
idRx	ID přijatého packetu	↓0	Long (I32)
idTxAck	ID posledního zpracovaného odeslaného packetu	↓0	Long (I32)
MORE	Příznak dalších dat v přijímacím mezibufferu		Bool
status	Indikátor vnitřního stavu		Long (I32)
	0 Žádná chyba		
	-1 Nepovedlo se otevřít port		
	1 Přetečení vysílacího buferu		
	2 Chyba odesílání dat		
	256 ... Chyba přijímání dat		

Parametr

port	Název komunikačního zařízení		String
baudrate	Bitová rychlost [bit/s] (0 = nenastavovat)	↓0 ↑4000000	Long (I32)
parity	Parita		Long (I32)
	0 Nenastaveno		
	1 Bez parity		
	2 Lichá parita		
	3 Sudá parita		
databits	Počet datových bitů (0 = nenastavovat)	↓0 ↑3	Long (I32)
stopbits	Počet stop bitů (0 = nenastavovat)	↓0 ↑2	Long (I32)
maxId	Maximální hodnota ID packetu	↓2 ↑10000000 ⊕4	Long (I32)
maxLen	Maximální délka přijatých dat	↓1 ↑10000000 ⊕64	Long (I32)
nmax	Rezervovaná paměť pro pole	↓8 ↑10000000 ⊕256	Long (I32)

Kapitola 17

LANG – Jazykové bloky

Obsah

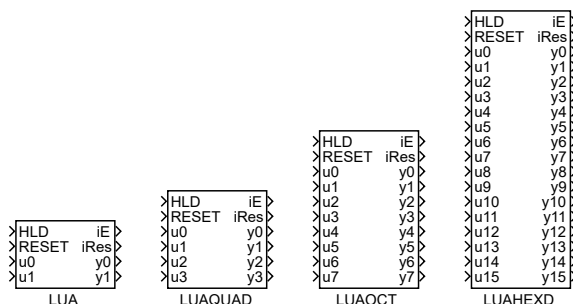
LUA, LUAQUAD, LUAOCT, LUAHEXD – Volně programovatelné bloky v Lua	620
PYTHON – Volně programovatelný blok v jazyce Python	624

Standardní funkční bloky systému REXYGEN pokrývají nejtypičtější potřeby v řídicích aplikacích. Nicméně stále existují situace, kdy je nutné (nebo pohodlnější) implementovat uživatelsky definovanou funkci. Pro tyto účely slouží bloky z knihovny LANG, případně blok [REXLANG](#).

LUA, LUAQUAD, LUAOCT, LUAHEXD – Volně programovatelné bloky v Lua

Symbody bloků

Licence: [REXLANG](#)



Popis funkce

Standardní funkční bloky systému REXYGEN pokrývají většinu typických potřeb v aplikacích řízení. Existují však situace, kdy je nutné (nebo vhodnější) implementovat uživatelsky definovanou funkci. Blok [REXLANG](#) pokrývá tento případ pro aplikace, kde je požadováno striktně real-time chování. Ve zbylých případech lze použít blok LUA.

Blok LUA implementuje uživatelsky definovaný algoritmus napsaný ve skriptovacím jazyce Lua a ve srovnání s blokem [REXLANG](#) nabízí lepší uživatelskou zkušenost ve fázi vývoje algoritmu a umožňuje rozšíření funkční sady systému REXYGEN prostřednictvím různých knihoven třetích stran dostupných v prostředí Lua.

K dispozici jsou celkem čtyři varianty bloku LUA s různými počty vstupů a výstupů. Vstupy a výstupy bloku jsou indexovány od 0 do 15. Vstupy a výstupy bloku mohou být libovolného datového typu, který je podporován systémem REXYGEN.

Skriptovací jazyk

Skriptovací jazyk je standardní Lua v5.4 (viz [18]). Každý blok odkazuje na skript uložený ve zdrojovém souboru *.lua nebo obsahuje skript přímo v parametru bloku. Skript může volitelně obsahovat funkce s vyhrazenými názvy, které jsou poté spouštěny systémem REXYGEN.

Funkce `main()` je spouštěna během běhu pravidelně. Kromě funkce `main()` je funkce `init()` spouštěna jednou při inicializaci a po resetu bloku. Funkce `exit()` je spouštěna jednou, když je řídicí algoritmus zastaven nebo před resetem bloku. Funkce `parchange()` je spouštěna při změně parametrů bloku v systému REXYGEN.

Dále je během kompilace konfigurace systému REXYGEN spuštěna funkce `compile()`. Tato funkce může být použita pro modifikaci tabulky `REX.ctx`, která je poté po kompilaci serializována a uložena v konfiguraci a později deserializována během inicializace bloku na cílovém zařízení. Tímto způsobem může skript vytvořit sdílený kontext mezi

kompilačním prostředím a cílovým zařízením. Jelikož je skript do jisté míry spouštěn jak na hostitelském zařízení, tak na cílovém zařízení, jsou skriptu k dispozici položky `REX.host` a `REX.target`, které umožňují rozlišit mezi těmito dvěma prostředími.

Skripty na cílovém zařízení

Lua interpreter může načíst skript ze specifikovaného souboru nebo přímo z řetězce. Pokud je parametr `embedded` nastaven na `on`, skript odkazovaný blokem v parametru `src` je zabudován do konfigurace systému REXYGEN během kompilačního procesu a bude načten během inicializace bloku, jakmile je konfigurace stažena a spuštěna na cílovém zařízení. V opačném případě je skript načten ze souboru na cílovém zařízení, který je opět specifikován parametrem `src`.

API pro výměnu dat

Pro účely výměny dat mezi interpretem Lua a systémem REXYGEN byl vyvinut objekt `REX`, který obsahuje API pro výměnu dat a je vkládán do každé instance interpreta bloku `LUA`.

I/O objekty

Rozsah vstupních a výstupních objektů závisí na použitém bloku. Nejkompaktnější blok `LUA` obsahuje 2 vstupy a 2 výstupy, `LUAQUAD` 4 vstupy a 4 výstupy, `LUAOCT` 8 vstupů a 8 výstupů a největší blok `LUAHEXD` 16 vstupů a 16 výstupů.

- `REX.u0` - `REX.u15`
 - objekty reprezentující *vstupy* bloku v prostředí Lua
- `REX.p0` - `REX.p15`
 - objekty reprezentující *parametry* bloku v prostředí Lua
- `REX.y0` - `REX.y15`
 - objekty reprezentující *výstupy* bloku v prostředí Lua

Přístup k hodnotám

I/O objekty musí mít specifikován svůj datový typ během kompilace konfigurace systému REXYGEN. Výchozí datový typ je `double`, ale lze jej změnit pomocí funkce `type()`.

Příklad nastavení datového typu I/O bloku:

```
REX.u0.type("int32")
REX.y0.type("string")
```

Všechny I/O objekty obsahují funkci `v()`, která provádí konverzi z datových typů systému REXYGEN do datových typů jazyka Lua. Hodnota pak může být uložena do proměnných a použita v algoritmu bloku.

Příklad čtení hodnoty vstupu bloku:

```
local x = REX.u0.v()

local myU0 = REX.u(0)
local x = myU0.v()
```

Nastavení hodnoty REXYGEN ze skriptu Lua lze provést funkcí `setV()`, která provádí konverzi z datových typů Lua do datových typů REXYGEN a exportuje hodnotu do odpovídajícího výstupu/parametru bloku.

Příklad zápisu hodnoty na výstup bloku:

```
REX.y0.setV(42)
```

```
local myY0 = REX.y(0)
myY0.setV(42)
```

Pole

Poli lze manipulovat prostřednictvím funkce `v()`, ale přímé konverze mezi poli systému REXYGEN a tabulkami prostředí jazyka Lua nejsou paměťově efektivní. Kromě toho vstupní objekty podporují přímý přístup k hodnotám pole systému REXYGEN připojeného ke vstupu bloku pomocí funkcí `arr()` a `setArr()`.

Příklad čtení a zápisu hodnoty vstupu bloku pro matici:

```
local x = REX.u0.arr(0, 0)
REX.u0.setArr(0, 0, 42)
```

Externí položky

Objekt REX obsahuje funkci `Item()`, která vrací odkaz na externí položku systému REXYGEN na základě řetězce připojení specifikovaného v parametru funkce.

Příklad vytvoření odkazu na externí položku a nastavení její hodnoty:

```
local cns = REX.Item("myproject_task.CNS:scv")
cns.setV("abc")
```

Trasování

Objekt REX obsahuje funkce `Trace`, `TraceError`, `TraceWarning`, `TraceVerbose` a `TraceInfo`, které lze použít pro zápis zpráv do systémového logu REXYGEN.

Příklad zápisu zprávy:

```
REX.Trace("abc")
```

Další funkce

`REX.RexDataPath()` – `RexDataPath()` je funkce, která vrací cestu ke složce s daty systému REXYGEN na dané platformě. To může být užitečné pro psaní platformově nezávislého kódu, který vyžaduje přístup k souborovému systému pomocí absolutních cest.

Tento blok nepropaguje kvalitu signálu. Více informací je uvedeno v sekci [1.4](#).

Vstup

HLD	Pozastavení	Bool
RESET	Reset bloku	Bool
u0..u15	Vstupní signál	Any

Parametr

<code>src</code>	Jméno souboru se skriptem nebo samotný skript	<code>⊙program.lua</code>	String
<code>embedded</code>	Uložení skriptu do programu exekutivy	<code>⊙on</code>	Bool
<code>limit</code>	Omezení vykonávání Lua interpreteru na periodu	<code>↓0</code>	Long (I32)
<code>p0..p15</code>	Parametr		Any

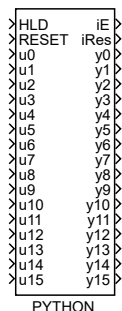
Výstup

<code>iE</code>	Kód chyby		Error
	<code>i</code> kód chyby REXYGENu		
<code>iRes</code>			Long (I32)
<code>y0..y15</code>	Výstupní signál		Any

PYTHON – Volně programovatelný blok v jazyce Python

Symbol bloku

Licence: [REXLANG](#)



Popis funkce

V některých případech se může stát, že je do řídicího algoritmu nutné implementovat funkci, kterou nelze efektivně vytvořit z dostupné množiny bloků. Pro takový účel byl vyvinut blok [REXLANG](#), který je vhodný zejména pro aplikace se striktními požadavky na běh programu v reálném čase. V opačném případě lze jako alternativu využít blok PYTHON.

Blok PYTHON implementuje algoritmus definovaný skriptovacím jazykem Python. V porovnání s blokem [REXLANG](#) nabízí snazší vývoj algoritmu a umožňuje rozšířit škálu funkcí poskytovaných systémem REXYGEN prostřednictvím balíčků a knihoven dostupných v ekosystému jazyka Python.

Upozornění: blok PYTHON je určen pro prototypování a experimentální aplikace, proto prosím zvažte velmi pečlivě jeho použití ve vaší aplikaci. PYTHON je experimentální blok a vždy bude. Existují situace, které mohou vést k neočekávanému chování nebo dokonce zamrznutí exekutivy. Balíčky mohou být špatně napsány nebo mohou mít špatně implementovanou reinicializaci a jejich použití může vést až k pádu. Pro tento blok poskytujeme pouze velmi omezenou podporu.

Skriptovací jazyk

Skriptovacím jazykem bloku je standardní Python 3 (viz [19]). Každý blok se odkazuje na skript napsaný v *.py zdrojovém souboru. Zdrojový kód může volitelně obsahovat funkce s danými jmény, které jsou spouštěny systémem REXYGEN.

Funkce `main()` je spouštěna periodicky za běhu systému REXYGEN. Krom funkce `main()` jsou spouštěny funkce `init()` při startu řídicího algoritmu a po resetu bloku, funkce `exit()` při skončení řídicího algoritmu a před resetem bloku a funkce `parchange()` při změně hodnot parametrů bloku.

Skripty na cílovém zařízení

Standardní interpret umožňuje načítání modulů/skriptů z různých lokací na cílovém zařízení. Blok `PYTHON` se může odkazovat na jakýkoliv skript dostupný standardnímu interpretu a navíc umožňuje načítání skriptů z adresáře `/rex/scripts/python`. Uživatelské skripty mohou být nahrávány přímo do této složky a nebo lze nastavit parametr `embedded` na hodnotu `on`. To způsobí, že referovaný skript je vložen do exekutivy při kompilaci a je následně dočasně extrahován do složky `/rex/scripts/python/embedded` při inicializaci bloku po nahrání a spuštění exekutivy na cílovém zařízení.

API pro výměnu dat

Pro výměnu dat mezi interpretrem jazyka Python a systémem REXYGEN byl vyvinut modul `PyRexExt` v podobě nativního rozšíření interpretu. Modul obsahuje objekt `REX`, který obsluhuje veškeré operace výměny dat. Pro inicializaci API lze použít následující úryvek kódu na začátku skriptu.

```
from PyRexExt import REX
```

I/O objekty

`REX.u0` - `REX.u15`

– objekty reprezentující *vstupy* bloku v prostředí jazyka Python

`REX.p0` - `REX.p15`

– objekty reprezentující *parametry* bloku v prostředí jazyka Python

`REX.y0` - `REX.y15`

– objekty reprezentující *výstupy* bloku v prostředí jazyka Python

Čtení a zápis hodnot

Všechny I/O objekty obsahují atribut `v`. Čtení atributu `v` provádí konverzi z datových typů systému REXYGEN na datové typy jazyka Python. Vracena hodnota pak může být uložena do proměnné a použita v algoritmu bloku. Pole systému REXYGEN jsou konvertovány na seznam hodnot v případě jednodimenzionálního pole, nebo na seznam seznamů v případě multidimenzionálního pole.

Příklad čtení hodnoty vstupu bloku:

```
x = REX.u0.v
```

Zápis atributu `v` naopak provádí konverzi z datových typů jazyka Python na datové typy systému REXYGEN a zápis hodnoty na příslušný výstup/parametr bloku.

Příklad zápisu hodnoty na výstup bloku:

```
REX.y0.v = 5
```

Pole

Objekty vstupů a výstupů obsahují atribut `size`. Tento atribut je jen pro čtení a vrací

objekt typu `tuple` s počtem řádek a počtem sloupců. S poli lze manipulovat prostřednictvím atributu `v`, ale tento přístup není příliš efektivní kvůli konverzi mezi poli a seznamy jazyk Python. Proto objekty vstupů a výstupů podporují operátor indexace `[]`, který provádí konverzi pouze dané položky v poli.

Příklad čtení vstupu bloku pro jednodimenzionální pole:

```
x = REX.u0[0]
```

Příklad zápisu hodnoty do multidimenzionálního pole výstupu bloku:

```
REX.u0[1, 3] = 5
```

Externí položky

Objekt `REX` obsahuje metodu `Item`, která vrací handle na požadovanou externí položku systému `REXYGEN` specifikovanou přípojným řetězcem předaným v prvním parametru metody.

Příklad vytvoření externí položky a nastavení její hodnoty:

```
cns = REX.Item("myproject_task.CNS:scv")
cns.v = "abc"
```

Trasování

Objekt `REXYGEN` obsahuje metody `Trace`, `TraceError`, `TraceWarning`, `TraceVerbose` and `TraceInfo`, které mohou být použity pro zápis zpráv do logu systému `REXYGEN`. Každá zpráva má k sobě připojený `stacktrace`.

Příklad logování zprávy:

```
REX.Trace("abc")
```

Dodatečné vlastnosti

`REX.RexDataPath` – `RexDataPath` je konstantní řetězec obsahující cestu k datovému adresáři systému `REXYGEN` na dané platformě. Tato vlastnost může být užitečná při psaní platformově nezávislého kódu manipulujícímu se souborovým systémem pomocí absolutních cest.

Definování datových typů

Pro správnou výměnu dat mezi systémem `REXYGEN` a prostředím jazyka Python musí být striktně definované datové typy vstupních signálů `u0..u15`, výstupních signálů `y0..y15` a parametrů `p0..p15`. Z toho důvodu musí být vytvořen konfigurační soubor pro každý skript se stejným názvem a s přidanou příponou `.cfg` (e.g. `program.py.cfg`). Pokud tento soubor chybí, je vytvořen při kompilaci projektu s tím, že všechny datové vstupy jsou nastaveny na defaultní hodnotu `double`. Nepředpokládá se, že by tento soubor byl editován přímo. Ke konfiguraci datových typů lze použít editor `REXYGEN` studia specifický pro blok `PYTHON`. Pro vstupy výstupy a parametry jsou dostupné datové typy `boolean`, `uint8`, `int16`, `uint16`, `int32`, `uint32`, `int64`, `float`, `double`, `string` a pro vstupy a výstupy jsou navíc dostupné datové typy `array`, `numpy` a `image`.

Pro tyto datové typy `numpy` a `image` musí být na cílovém zařízení nainstalován balíček `numpy`. Pro vstupy typu `numpy` je předpokládáno, že vstupní signál je typu pole, které je následně překonvertováno na nativní reprezentaci objektu `numpy`. Pro vstupy typu `image` je předpokládáno, že vstupní signál je datového typu `image` z modulu `RexVision`, který je rovněž překonvertován na nativní reprezentaci typu `numpy` a může tedy být přímo použitý s `OpenCV` balíčkem jazyka Python.

Pro výstupy datového typu `numpy` je předpokládáno, že budou nastaveny ve skriptu pomocí objektu typu `numpy`, který je následně překonvertován na běžné pole. Pro výstupy datového typu `image` je předpokládáno, že budou nastaveny ve skriptu pomocí objektu typu `numpy`, který je následně překonvertován na objekt typu `image` definovaný modulem `RexVision`.

Příklad definice datových typů

Následující příklad ukazuje zkrácenou verzi souboru ve formátu `JSON`, který popisuje datové typy vstupů, výstupů a parametrů bloku.

```
{
  "types": {
    "in": [
      {
        "idx": 0,
        "type": "double"
      },
      . . .
      {
        "idx": 15,
        "type": "double"
      }
    ],
    "param": [
      {
        "idx": 0,
        "type": "double"
      },
      . . .
      {
        "idx": 15,
        "type": "double"
      }
    ],
    "out": [
      {
        "idx": 0,
        "type": "double"
      },
      . . .
      {
        "idx": 15,
        "type": "double"
      }
    ]
  }
}
```

```
    }
}
```

Příklad skriptu v jazyce Python

Následující příklad ukazuje jednoduchý zdrojový kód v jazyce Python, který sčítá dva vstupní signály a dva uživatelem definované parametry.

```
from PyRexExt import REX

def main():
    REX.y0.v = REX.u0.v + REX.u1.v
    REX.y1.v = REX.p0.v + REX.p1.v
    return
```

Instalace - Debian

Prostředí jazyka Python by mělo být korektně nastaveno po instalaci debian balíčku `PythonBlk_T`. Pro instalaci s volitelnými balíčky `numpy` a `OpenCV` je třeba spustit následující příkazy z terminálu.

```
sudo apt install rex-pythonblk
sudo apt install python3-numpy python3-opencv
```

Instalace - Windows

Pro instalaci správné verze prostředí Python je doporučeno stáhnout a nainstalovat 64-bitovou verzi z [oficiálního repozitáře](https://www.python.org/ftp/python/3.9.6/) (<https://www.python.org/ftp/python/3.9.6/>). Při instalaci je vhodné se ujistit, že je zvolena možnost instalace programu `pip` a že cesta k binárním souborům interpretru bude přidána do systémové proměnné `PATH`.

Pro instalaci volitelných balíčků `numpy` a `OpenCV` je třeba spustit následující příkazy z příkazové řádky.

```
pip install numpy
pip install opencv-python
```

Omezení

Vzhledem k omezením zakořeněným v implementaci standardního interpretru jazyka Python není doporučeno používat více instancí bloku `PYTHON` na rozdílných úrovních exekutivy. Takové použití může vést k nepředvídatelnému chování a nestabilitě programu `RexCore`.

Tento blok nepropaguje kvalitu signálu. Více informací je uvedeno v sekci [1.4](#).

Vstup

HLD	Pozastavení	Bool
RESET	Reset bloku	Bool
u0..u15	Vstupní signál	Any

Parametr

<code>srcname</code>	Jméno souboru se skriptem	⊙ <code>program.py</code>	String
<code>embedded</code>	Uložení skriptu do programu exekutivy	⊙ <code>on</code>	Bool
<code>limit</code>	Omezení vykonávání Python interpretu na periodu	↓1 ⊙10000	Long (I32)
<code>limitMode</code>	Režim omezení vykonávání Python interpretu		Long (I32)
	0 bez omezení		
	1 počet instrukcí za periodu, po přerušení interpreter pokračuje dál		
	2 počet instrukcí za periodu, po přerušení interpreter začíná znovu		
	3 počet milisekund za periodu, po přerušení interpreter pokračuje dál		
	4 počet milisekund za periodu, po přerušení interpreter začíná znovu		
<code>p0..p15</code>	Parametr		Any

Výstup

<code>iE</code>	Kód chyby	Error
	i kód chyby REXYGENu	
<code>iRes</code>		Long (I32)
<code>DONE</code>	Příznak skončení vykonávání skriptu	Bool
<code>y0..y15</code>	Výstupní signál	Any

Kapitola 18

DSP – Zpracování speciálních signálů

Obsah

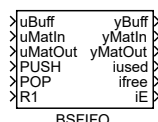
BSFIFO – Binární Struktura - serializace a deserializace do cyklického bufferu	632
BSGET, BSGETOCT – Binární struktura - získání hodnoty daného typ	634
BSGETV, BSGETOCTV – Binární struktura - získání pole hodnot daného typu	636
BSSET, BSSETOCT – Binární struktura - nastavení hodnoty daného typu	637
BSSETV, BSSETOCTV – Binární struktura - nastavení pole hodnot daného typu	638
FFT – Rychlá Fourierova transformace	639
MOSS – Přesný senzor pohybu	641
PCI – Přístup k paměti přes sběrnici PCI	643
PSD – Spektrální výkonová hustota	644
RAWCOPY – Kopírování vektoru po 1, 2 nebo 4 Bajtech	650

Knihovna DSP je přizpůsobena pro pokročilé zpracování digitálních signálů. Obsahuje bloky jako **FFT** pro operace rychlé Fourierovy transformace a **PSD** pro analýzu spektrální hustoty výkonu. Knihovna také nabízí **BSFIFO**, **BSGET**, **BSGETV**, **BSSET** a **BSSETV** pro ukládání do bufferu a jeho načítání, což umožňuje efektivní manipulaci s daty při úlohách zpracování signálů. Kromě toho knihovna obsahuje blok **MOSS** - pokročilý filtr pro inkrementální senzory. Tato kolekce bloků je nezbytná pro sofistikovanou analýzu a manipulaci se signály v digitálních systémech.

BSFIFO – Binární Struktura - serializace a deserializace do cyklického bufferu

Symbol bloku

Licence: [ADVANCED](#)



Popis funkce

Tento blok postupně přidává a vybírá data do velkého bufferu (přivedeného na vstup **uBuff**). Základní jednotka v bufferu je sloupec. Všechny matice (tj. matice nebo vektory přivedené na vstupy **uBuff**, **uMatIn**, **uMatOut**) musí mít stejnou velikost sloupce v bajtech. Data jsou organizována buď jako fronta (pokud **REV=off**) nebo jako zásobník (pokud **REV=on**). Chování bloku závisí na vstupech tímto způsobem:

- Pokud je **PUSH=on**, vkládá se do bufferu obsah matice **uMatIn** (všechny definované sloupce).
- Pokud je **POP=on**, vyjímá se z bufferu počet sloupců určených parametrem **col** a tato data se vloží do matice **uMatOut** (musí mít dostatečnou velikost).
- Pokud je **R1=on**, data se znovu načtou (hlavně počet platných sloupců) do bufferu bloku. Vlastní data se předávají odkazem a jsou proto sdílená. Tento signál je prioritní a blokuje signály **PUSH**, **POP**.

Chybové stavy (např. neodpovídající si rozměry matic, nedostatečné místo v některé matici, nedostatek dat v bufferu) jsou indikovány na výstupu **iE** a zprávou v **SystemLog**.

Tento blok nepropaguje kvalitu signálu. Více informací je uvedeno v sekci [1.4](#).

Vstup

uBuff	Vstupní reference na binární strukturu	Reference
uMatIn	Vstupní reference na matici nebo vektor	Reference
uMatOut	Vstupní reference na matici nebo vektor	Reference
PUSH	Povolení vkládání do bufferu	Bool
POP	Povolení vyčítání z bufferu	Bool
R1	Reset bloku	Bool

Parametr

OW	Režim přepisu nejstarších položek	Bool

REV	Vyčítání posledních vložených položek nejdříve	Bool
col	Počet vyčítaných sloupců	⊙1 Long (I32)

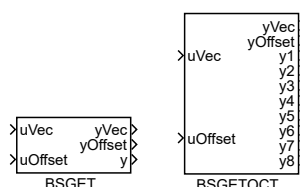
Výstup

yBuff	Výstupní reference na binární strukturu	Reference
yMatIn	Výstupní reference na matici nebo vektor	Reference
yMatOut	Výstupní reference na matici nebo vektor	Reference
iused	Počet použitých bajtů v bufferu	Long (I32)
ifree	Počet volných bajtů ve bufferu	Long (I32)
iE	Kód chyby	Error

BSGET, BSGETOCT – Binární struktura - získání hodnoty daného typ

Symbole bloků

Licence: [ADVANCED](#)



Popis funkce

Tato skupina bloků slouží pro získávání hodnot z binární struktury (bajtového pole). Pro zápis do binární struktury lze použít bloky [BSSET](#) a [BSSETOCT](#).

Pokud binární struktury přijdou po komunikaci, je možné je zpracovat přímo v bloku zprostředkovávajícím komunikaci. Typicky se jedná o programovatelný blok [REXLANG](#) nebo [PYTHON](#). Pomocí struktur je však možné předávat data i v rámci aplikace REXYGEN. Binární struktura se přivede ve formě pole (vektoru) bajtů na vstup `uVec`. Vstup `uOffset` udává posunutí (v bajtech) požadované hodnoty od začátku struktury. Typ hodnoty udává parametr `type`.

Výstup `yOffset` je začátek následujícího prvku ve struktuře. To je výhodné pro řetězení: pokud struktura obsahuje několik prvků za sebou, je možné zapojit vstup `uOffset` na výstup `yOffset` předchozího bloku a není nutné posunutí dopočítávat.

Bloky se liší jen v tom, že `BSGET` získává jednu hodnotu. Blok `BSGETOCT` je schopný získat až 8 hodnot (počet určuje parametr `m`).

Tento blok nepropaguje kvalitu signálu. Více informací je uvedeno v sekci [1.4](#).

Vstup

<code>uVec</code>	Vstupní reference na binární strukturu	Reference
<code>uOffset</code>	Posunutí (v bajtech) dat od začátku struktury	Long (I32)

Parametr

<code>BE</code>	Big-Endian pořadí bajtů (výchozí je Little-Endian, tj. Intel)	Bool
<code>m</code>	Počet aktivních položek	↓1 ↑8 ⊖8 Long (I32)
<code>type1..type8</code>	Datový typ hodnoty	↓2 ↑10 ⊖2 Long (I32)

Výstup

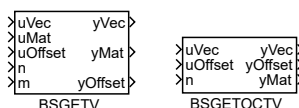
<code>yVec</code>	Výstupní reference na binární strukturu	Reference

<code>y0ffset</code>	Posunutí (v bajtech) dat od začátku struktury další hodnoty	<code>Long</code> (I32)
<code>y1..y8</code>	Požadovaná hodnota ze struktury	<code>Double</code> (F64)

BSGETV, BSGETOCTV – Binární struktura - získání pole hodnot daného typu

Symbole bloků

Licence: [ADVANCED](#)



Popis funkce

Tato skupina bloků slouží pro získávání hodnot z binární struktury (bajtového pole). Pro zápis do binární struktury lze použít bloky [BSSETV](#) a [BSSETOCTV](#).

Význam většiny parametrů je stejný jako u bloku [BSGET](#), ale tyto bloky načítají několik hodnot stejného typu a ukládají je do pole (matice). Matice má vždy *m* řádek a *n* sloupců. U bloku [BSGETV](#) jsou všechny prvky stejného typu (určuje parametr *type*) a data se vyplňují do matice přivedené na vstup *uMat*. Blok [BSGETOCTV](#) načítá až 8 vektorů. Každý řádek matice může být jiného typu. Blok alokuje matici sám. Matice je na výstupu *yMat*.

Tento blok nepropaguje kvalitu signálu. Více informací je uvedeno v sekci [1.4](#).

Vstup

<i>uVec</i>	Vstupní reference na binární strukturu	Reference
<i>uOffset</i>	Posunutí (v bajtech) dat od začátku struktury	Long (I32)
<i>n</i>	Počet sloupců matice	Long (I32)

Parametr

<i>BE</i>	Big-Endian pořadí bajtů (výchozí je Little-Endian, tj. Intel)	Bool
<i>m</i>	Počet aktivních položek	↓1 ↑8 ⊙8 Long (I32)
<i>nmax</i>	Rezervovaná paměť pro výstupní matici (celkový počet prvků)	Long (I32)
		↓1 ⊙32
<i>type</i>	Datový typ hodnoty	↓2 ↑10 ⊙2 Long (I32)
<i>type1..type8</i>	Datový typ hodnoty	↓2 ↑10 ⊙2 Long (I32)

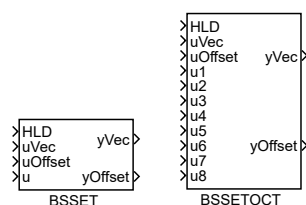
Výstup

<i>yVec</i>	Výstupní reference na binární strukturu	Reference
<i>yMat</i>		Reference
<i>yOffset</i>	Posunutí (v bajtech) dat od začátku struktury další hodnoty	Long (I32)

BSSET, BSSETOCT – Binární struktura - nastavení hodnoty daného typu

Symbyoly bloků

Licence: [ADVANCED](#)



Popis funkce

Tato skupina bloků slouží pro nastavování hodnot do binární struktury (bajtového pole). Funkce je inverzní bloku [BSGET](#) a [BSGETOCT](#), tj. všechny signály mají stejný význam, jen data se kopírují opačným směrem - ze vstupu **u** do binární struktury reprezentované polem bajtů připojeným ke vstupu **uVec**. Blok modifikuje binární strukturu jen pokud je **HLD=off**.

Tento blok nepropaguje kvalitu signálu. Více informací je uvedeno v sekci [1.4](#).

Vstup

HLD	Pozastavení	Bool
uVec	Vstupní reference na binární strukturu	Reference
uOffset	Posunutí (v bajtech) dat od začátku struktury	Long (I32)
u1..u8	Požadovaná hodnota do struktury	Double (F64)

Parametr

m	Počet aktivních položek	↓1 ↑8 ⊙8	Long (I32)
BE	Big-Endian pořadí bajtů (výchozí je Little-Endian, tj. Intel)		Bool
type1..type8	Datový typ hodnoty	↓2 ↑10 ⊙2	Long (I32)

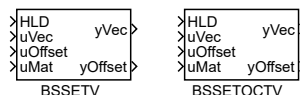
Výstup

yVec	Výstupní reference na binární strukturu	Reference
yOffset	Posunutí (v bajtech) dat od začátku struktury další hodnoty	Long (I32)

BSSETV, BSSETOCTV – Binární struktura - nastavení pole hodnot daného typu

Symbole bloků

Licence: [ADVANCED](#)



Popis funkce

Tato skupina bloků slouží pro nastavení matice hodnot do binární struktury (bajtového pole). Funkce je inverzní bloku [BSGETV](#) a [BSGETOCTV](#), tj. všechny signály mají stejný význam, jen data se kopírují opačným směrem - ze vstupu **uMat** do binární struktury reprezentované polem bajtů připojeným ke vstupu **uVec**. Blok modifikuje binární strukturu jen pokud je **HLD=off**.

Na rozdíl od bloku [BSGETV](#) se počet řádek a sloupců nezadáva, ale určuje se ze skutečné velikosti matice připojené na vstup **uMat**.

Tento blok nepropaguje kvalitu signálu. Více informací je uvedeno v sekci [1.4](#).

Vstup

uVec	Vstupní reference na binární strukturu	Reference
HLD	Pozastavení	Bool
uMat	Matice hodnot	Reference
uOffset	Posunutí (v bajtech) dat od začátku struktury	Long (I32)

Parametr

m	Počet aktivních položek	↓1 ↑8 ⊙8	Long (I32)
BE	Big-Endian pořadí bajtů (výchozí je Little-Endian, tj. Intel)		Bool
type1..type8	Datový typ hodnoty	↓2 ↑10 ⊙2	Long (I32)

Výstup

yVec	Výstupní reference na binární strukturu	Reference
yOffset	Posunutí (v bajtech) dat od začátku struktury další hodnoty	Long (I32)

FFT – Rychlá Fourierova transformace

Symbol bloku

Licence: [ADVANCED](#)



Popis funkce

Funkční blok **FFT** (Fast Fourier Transform) slouží pro výpočet rychlé Fourierovy transformace pomocí balíku PocketFFT [20]. PocketFFT implementovaný v C99 vychází z balíku FFTPack (ve Fortranu) [21], který byl vyvinut na základě kapitoly z [22].

Vstup **uc** odkazuje na vstupní data (vektor nebo matice) tohoto bloku. Pokud vstup **uc** odkazuje na sloupcový vektor (počet sloupců $m = 1$) s počtem prvků **n**, pak bude FFT počítána z jednoho signálu s **n** vzorky. Pokud vstup **uc** odkazuje na matici s **n** řádky a **m** sloupci, pak bude FFT počítána **m** krát (jednou pro každý sloupec).

Vstupní data jsou zpracovávána podle parametru **mode** (viz níže), který určuje, zda bude FFT počítána pro reálná nebo komplexní data a přímou nebo zpětnou transformaci. Přesněji, parametr **mode** určuje, jaká z funkcí **rfft_forward()**, **rfft_backward()**, **cfft_forward()** nebo **cfft_backward()** z balíku PocketFFT bude volána.

Vstup **uf** odkazuje na výstupní data, která mají stejný počet **n** řádků a **m** sloupců jako data, na něž odkazuje **uc**. Pokud je vstup **uf** připojen na předalokovaný vektor/matici pak na výstupní data FFT algoritmu odkazuje výstup **yf**. Pokud není vstup **uf** připojen, počítá se FFT na místě a výstupní data jsou ukládána v poli, na něž se odkazuje výstup **yc**, a tedy data odkazovaná vstupem **uc** jsou přepisována.

Odkaz **uc** je kopírován na **yc**, odkaz **uf** je kopírován na **yf**.

Vstup **HLD** umožňuje dočasně zastavit výpočet FFT.

Tento blok nepropaguje kvalitu signálu. Více informací je uvedeno v sekci 1.4.

Vstup

uc	Vstupní odkaz na vstupní/výstupní data	Reference
uf	Vstupní odkaz na volitelná výstupní data	Reference
HLD	Pozastavení	Bool

Parametr

mode	Výpočetní mód FFT	⊙1 Long (I32)
1	 Reálná dopředná	
2	 Reálná zpětná	
3	 Komplexní dopředná	
4	 Komplexní zpětná	

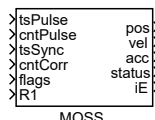
Výstup

<code>yc</code>	Výstupní odkaz na vstupní/výstupní data	<code>Reference</code>
<code>yf</code>	Výstupní odkaz na volitelná výstupní data	<code>Reference</code>
<code>E</code>	Příznak chyby	<code>Bool</code>

MOSS – Přesný senzor pohybu

Symbol bloku

Licence: [ADVANCED](#)



Popis funkce

Blok **MOSS** implementuje pokročilý filtr pro inkrementální (kvadraturní) senzory polohy. Blok vyžaduje speciální hardware, neboť pro správnou činnost je nutné znát nejen aktuální hodnotu z čidla, ale také časovou značku posledního pulsu, směr pohybu při posledním pulzu, časovou značku referenčního okamžiku (ze stejného zdroje jako časová značka pulzu). Výstupem bloku je nejen filtrovaná poloha, ale také rychlost a zrychlení. Pro správnou činnost je potřeba vhodně zvolit parametr **alpha**. Menší hodnota snižuje šum, ale zvětšuje zpoždění signálu.

Pokud není zaznamenán nový pulz po dobu **stall**, je senzor považován za zastavený a výstupy **pos**, **vel**, **acc** jsou nastaveny na 0. Je-li **pos** > **maxpos**, je vnitřní pozice zpracovaná Kalmanovým filtrem dekrementována o celočíselný násobek **maxpos** a inkrementována zpět pro výstup. To způsobí, že algoritmus filtru vypočítá dostatečně malá čísla a nesníží přesnost kvůli chybám zaokrouhlování. Výchozí hodnota by se normálně neměla měnit. Pokud není po dlouhou dobu přijat žádný impuls, výstup prediktora začne driftovat. Pro překonání tohoto driftu, pokud není detekován žádný pulz po dobu delší než **mindivert**, je výstupní pozice změněna o $\pm$ puls od vstupní (měřené) pozice.

Poznámka 1: Zdánlivě nelze polohu určit přesněji než je kvantizační chyba (± 1 pulz) měření, ale znalost rychlosti umožňuje odhadovat polohu přesněji. Dále se může zdát, že nelze určit rychlost přesněji než jako podíl počtu pulzů a rozdílu časových značek, ale vložení obou signálů do grafu je vidět, že **MOSS** signál zlepšuje. Určování zrychlení diferencí vede na naprosto nepoužitelné hodnoty.

Poznámka 2: Interně blok realizuje Kalmanův filtr pro systém se 2 integrátory (tj. vstup zrychlení, výstup poloha). Filtr je diskretizován v každé periodě znova, přičemž diskretizační perioda je aktuální rozdíl časových značek. Pro odvození filtru je potřeba znát vstupní a výstupní poruchy. Pokud uvažujeme bílý (gaussovský) šum, stačí znát jejich poměr vyjádřený parametrem $\alpha = \ln(Q/R)$, kde Q je rozptyl na vstupu a R je rozptyl měření výstupu.

Poznámka 3: Kalmanův filtr v tomto případě vede na filtr typu dolní propust (3. řád, ale jen 6 dB/okt. s rezonančním převýšením) s mezní frekvencí ω_R , přičemž $\alpha = 6 \cdot \ln(\omega_R)$.

Poznámka 4: Filtr v tomto bloku je stejný jako v bloku **KDER** s tím, že v bloku **MOSS** je napevno **norder** = 3. Počítají se jen druhé derivace (v **KDER** až pět derivací), ale výpočet funguje správně pro neekvidistantní vzorkování (tak, jak hodnoty přicházejí z

kvadraturního enkodéru nebo počítadla zubů).

Tento blok nepropaguje kvalitu signálu. Více informací je uvedeno v sekci 1.4.

Vstup

tsPulse	Časová značka posledního pulsu	DWord (U32)
cntPulse	Poslední stav čítače pulsů	DWord (U32)
tsSync	Časová značka synchronizačního pulsu	DWord (U32)
cntCorr	Odchylka aktuálního pulsu od ideální polohy	Double (F64)
flags	Vstupní stavové příznaky (1: POS, 2: NEG, 4: RUN)	DWord (U32)
R1	Reset bloku	DWord (U32)

Parametr

freq	Frekvence zdroje časové značky [Hz]	↓0.0 ⊕100000000.0	Double (F64)
stall	Čas pro aktivaci zastavení [s]	↓0.0 ⊕0.08	Double (F64)
alpha	Návrhový parametr Kalmanova filtru	↓0.0 ↑200.0 ⊕26.0	Double (F64)
maxpos	Optimalizace zaokrouhlovacích chyb Kalmanova filtru	↓0.0 ⊕1e+10	Double (F64)
mindivert	Čas pro aktivaci omezení odchylky prediktoru [s]	↓0.0 ⊕0.003	Double (F64)

Výstup

pos	Filtrovaná poloha	Double (F64)
vel	Filtrovaná rychlost	Double (F64)
acc	Filtrované zrychlení	Double (F64)
status	Výstupní stavové příznaky (1: POS, 2: NEG, 4: RUN, 8: INIT, 16: PULSE, 32: STALLED, 64: DIVERT)	Long (I32)
iE	Kód chyby	Error

PCI – Přístup k paměti přes sběrnici PCI

Symbol bloku

Licence: [ADVANCED](#)



Popis funkce

Blok PCI poskytuje přístup k zdrojům PCI zařízení definovaným cestou:

`/sys/bus/pci/devices/<device>/resource<resource>`

Data jsou na výstupu `y` bloku prezentována jako reference na pole bajtů. Zařízení se specifikuje pomocí parametru `pci_device`, který přijímá dva formáty:

- Slot zařízení podle formátu výpisu příkazu `lspci`, tedy:
`<domain>:<bus>:<slot>.<func>`.
- Kombinaci identifikátorů výrobce a zařízení s možným indexem, ve formátu:
`[<vendor>]:[<device>][#<index>]`.

Identifikátory jsou hexadecimální čísla získaná z příkazu `lspci -nn` bez předřazeného "0x". Může být vynechán identifikátor výrobce nebo zařízení. Bez specifikovaného indexu se výchozí volbou stává první nalezené zařízení odpovídající zadaným identifikátorům.

Výchozí nastavení parametru `size` je 0, což znamená použití celého zdroje. To však může vést k alokaci velké oblasti virtuální paměti.

Poznámka: Blok je podporován pouze na platformách s operačním systémem Linux.

Tento blok nepropaguje kvalitu signálu. Více informací je uvedeno v sekci [1.4](#).

Parametr

<code>pci_device</code>	PCI zařízení: <code><domain>:<bus>:<slot>.<func></code> nebo <code>id</code>	<code>String</code>
	<code>[<vendor>]:[<device>][#<index>]</code>	
<code>resource</code>	Číslo pci zdroje	<code>Long (I32)</code>
<code>offset</code>	Offset v pci zdroji	<code>Large (I64)</code>
<code>size</code>	Velikost dat k otevření (0 pro celý zdroj)	<code>Large (I64)</code>

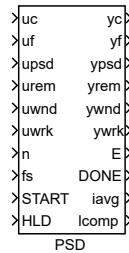
Výstup

<code>y</code>	Odkaz na pole bajtů	<code>Reference</code>

PSD – Spektrální výkonová hustota

Symbol bloku

Licence: [ADVANCED](#)



Popis funkce

Funkční blok PSD (Power Spectral Density) slouží pro výpočet výkonové spektrální hustoty (PSD, periodogramu) pomocí balíku PocketFFT [20]. PocketFFT implementovaný v C99 vychází z balíku FFTPack (ve Fortranu) [21], který byl vyvinut na základě kapitoly z [22].

PSD se nejčastěji počítá ze segmentů délky N vzorků vstupního signálu. Data těchto segmentů jsou vynásobena váhovou funkcí (window function), též nazývanou oknem (např. Hammingovo okno). Z takto vážených dat je vypočtena rychlá Fourierova transformace (FFT), a z ní pak spektrální výkonová hustota (PSD) každého segmentu. Nakonec se zprůměruje `navg` těchto výsledků (viz parametr `navg`). Vstupní signál je rozdělen do segmentů následujících po sobě nebo se segmenty vždy o polovinu překrývají (Welchova metoda [23], [24]). Výsledná PSD může být vypočtena v decibelech (dB), pokud je parametr `DB = on`.

Vstup `uc` odkazuje na vstupní data (vektor nebo matice) tohoto bloku. Pokud vstup `uc` odkazuje na sloupcový vektor (počet sloupců $m = 1$) s počtem prvků n , pak bude PSD počítána z jednoho signálu s n vzorky. Pokud vstup `uc` odkazuje na matici s n řádky a m sloupci, pak bude PSD počítána m krát (jednou pro každý sloupec). Pokud je vstup $n > 0$, pak bude počet vzorků segmentu $N = n$, jinak bude N nastaveno na počet sloupců pole, na něž odkazuje vstup `uc`.

Vstup `uf` odkazuje na interní vektor dimenze $\geq N$ pro výpočet FFT. Vstup `upsd` odkazuje na výslednou PSD s `npsd` řádky a m sloupci. Počet frekvencí PSD $npsd = N/2 + 1$ pro sudé N nebo $npsd = (N + 1)/2$ pro liché N . Vstup `uwrk` odkazuje na pracovní pole se stejnými rozměry jako pole odkazované vstupem `upsd`. Pokud vstupní data obsahují víc prvků než může být zpracováno v jednom spuštění bloku PSD, jsou zbývající data zkopírována do pole odkazovaného vstupem `urem` naalokovaného s N řádky a m sloupci.

Zpracování vstupních dat je určeno parametry `mode` a `mtrig` (viz níže). Parametr `mode` určuje způsob výpočtu PSD, podle toho, zda vstupní data odkazovaná vstupem `uc` jsou reálná nebo komplexní a zda bude použito překrývání segmentů o polovinu. Vstupní data vzorkovaná s frekvencí připojenou na vstup `fs` mohou být získávána v jiné

úloze (např. s podstatně kratší periodou vzorkování) než v té, ve které je počítána PSD. Způsob spouštění výpočtu PSD je určen parametrem **mtrig** (trigger mode). Výpočet PSD může být spouštěn každou periodu běhu bloku PSD, v případě, že ve vstupním poli je dostatečný počet vzorků (alespoň rovný hodnotě N), nebo náběžnou hranou vstupu **START**.

Na vstup **uwnd** je připojen vektor dimenze N , do kterého se uloží hodnoty váhové funkce okna, zvoleného parametrem **iwin**. Váhové funkce okna mohou mít (volitelně) celočíselný parametr **lwnd** a/nebo reálný parametr **rwnd**. Některá okna lze používat ve dvou variantách:

Symetrické (pro **lwnd**=0). Je vypočteno okno délky $L = N$ vzorků, první a poslední vzorek je shodný. Tato varianta je vhodná pro návrh filtrů s konečnou impulzní odezvou (Finite Impulse Response, FIR).

Periodické (pro **lwnd**=1). Je vypočteno okno délky $L = N + 1$ vzorků, přičemž se uloží prvních N vzorků. Periodická verze se přednostně používá pro spektrální analýzu, protože diskrétní Fourierova transformace předpokládá periodický průběh vstupního vektoru.

Váhové funkce oken jsou definovány následujícími výrazy:

iwin=1 Bartlettovo-Hahnovo okno (Bartlett-Hahn Window)

$$w(n) = 0.62 - 0.48 \left| \left(\frac{n}{N-1} - 0.5 \right) \right| + 0.38 \cos \left[2\pi \left(\frac{n}{N-1} - 0.5 \right) \right], \quad 0 \leq n \leq N-1.$$

iwin=2 Bartlettovo okno (Bartlett Window)

$$w(n) = \begin{cases} \frac{2n}{N}, & 0 \leq n \leq \frac{N}{2}, \\ 2 - \frac{2n}{N}, & \frac{N}{2} \leq n \leq N. \end{cases}$$

iwin=3 Blackmanovo okno (Blackman Window)

$$w(n) = 0.42 - 0.5 \cos \left(\frac{2\pi n}{L-1} \right) + 0.08 \cos \left(\frac{4\pi n}{L-1} \right), \quad 0 \leq n \leq M-1,$$

kde M je $N/2$ pro sudé N a $(N+1)/2$ pro liché N .

iwin=4 Blackmanovo-Harrisovo okno (Blackman-Harris Window)

$$w(n) = a_0 - a_1 \cos \left(\frac{2\pi n}{L-1} \right) + a_2 \cos \left(\frac{4\pi n}{L-1} \right) - a_3 \cos \left(\frac{6\pi n}{L-1} \right), \quad 0 \leq n \leq N-1$$

kde $a_0 = 0.35875$, $a_1 = 0.48829$, $a_2 = 0.14128$ a $a_3 = 0.01168$.

iwin=5 **Bohmanovo okno** (Bohman Window)

$$w(x) = (1 - |x|) \cos(\pi |x|) + \frac{1}{\pi} \sin(\pi |x|), \quad -1 \leq x \leq 1$$

iwin=6 **Čebyševovo okno** (Chebyshev Window)

Použitím Čebyševových polynomů n -tého řádu

$$T_n(x) \triangleq \begin{cases} \cos[n \cos^{-1}(x)], & |x| \leq 1, \\ \cosh[n \cosh^{-1}(x)], & |x| > 1, \end{cases}$$

je Fourierova transformace Čebyševova okna

$$W(k) = \frac{T_{N-1}[x_0 \cos(\pi k/N)]}{T_{N-1}(x_0)}$$

kde

$$x_0 = \cosh \left[\frac{\cosh^{-1}(10^{r/20})}{N-1} \right].$$

iwin=7 **Okno s plochým vrcholem** (Flat Top Window)

$$w(n) = a_0 - a_1 \cos\left(\frac{2\pi n}{L-1}\right) + a_2 \cos\left(\frac{4\pi n}{L-1}\right) - a_3 \cos\left(\frac{6\pi n}{L-1}\right) + a_4 \cos\left(\frac{8\pi n}{L-1}\right),$$

kde $0 \leq n \leq N-1$ and $a_0 = 0.21557895$, $a_1 = 0.41663158$, $a_2 = 0.277263158$, $a_3 = 0.083578947$ a $a_4 = 0.006947368$.

iwin=8 **Gaussovo okno** (Gauss Window)

$$w(n) = e^{-\frac{1}{2} \left[\alpha \frac{n}{(N-1)/2} \right]^2} = e^{-n^2/(2\sigma^2)}$$

kde $-(N-1)/2 \leq n \leq (N-1)/2$ a $\sigma = (N-1)/(2\alpha)$ je směrodatná odchylka Gaussovy funkce hustoty pravděpodobnosti.

iwin=9 **Hammingovo okno** (Hamming Window)

$$w(n) = 0.54 - 0.46 \cos\left(2\pi \frac{n}{L-1}\right), \quad 0 \leq n \leq N-1,$$

iwin=10 **Hannovo okno** (Hann Window)

$$w(n) = 0.5 \left[1 - \cos\left(2\pi \frac{n}{L-1}\right) \right], \quad 0 \leq n \leq N-1,$$

iwin=12 Nuttallova varianta Blackmanova-Harrisova okna (Nuttall's Blackman-Harris Window)

$$w(n) = a_0 - a_1 \cos\left(\frac{2\pi n}{L-1}\right) + a_2 \cos\left(\frac{4\pi n}{L-1}\right) - a_3 \cos\left(\frac{6\pi n}{L-1}\right), \quad 0 \leq n \leq N-1$$

kde $a_0 = 0.3635819$, $a_1 = 0.4891775$, $a_2 = 0.1365995$ a $a_3 = 0.0106411$.

iwin=13 Parzenovo okno (Parzen Window)

$$w(n) = \begin{cases} 1 - 6 \left(\frac{|n|}{N/2}\right)^2 + 6 \left(\frac{|n|}{N/2}\right)^3, & 0 \leq |n| \leq (N-1)/4, \\ 2 \left(1 - \frac{|n|}{N/2}\right)^3, & (N-1)/4 < |n| \leq (N-1)/2. \end{cases}$$

iwin=14 Obdélníkové okno (Rectangular Window)

$$w(n) = 1, \quad 0 \leq n \leq N-1.$$

Tento blok neprovádí násobení všech datových vzorků hodnotou 1, a proto nemusí být vstup **uwnd** připojen (vektor okna se nepoužívá).

iwin=16 Trojúhelníkové okno (Triangular Window)

Pro liché N :

$$w(n) = \begin{cases} \frac{2n+2}{N+1}, & 0 \leq n \leq (N-1)/2, \\ 2 - \frac{2n+2}{N+1}, & (N-1)/2 < n \leq (N-1). \end{cases}$$

Pro sudé N :

$$w(n) = \begin{cases} \frac{2n+1}{N}, & 0 \leq n \leq N/2 - 1, \\ 2 - \frac{2n+1}{N}, & N/2 \leq n \leq (N-1). \end{cases}$$

iwin=17 Tukeyovo okno (Tukey Window)

$$w(x) = \begin{cases} \frac{1}{2} \left\{ 1 + \cos\left(\frac{2\pi}{r} [x - r/2]\right) \right\}, & 0 \leq x < \frac{r}{2}, \\ 1, & \frac{r}{2} \leq x < 1 - \frac{r}{2}, \\ \frac{1}{2} \left\{ 1 + \cos\left(\frac{2\pi}{r} [x - 1 + r/2]\right) \right\}, & 1 - \frac{r}{2} \leq x < 1, \end{cases}$$

kde r je parametrem **rwnd**, s implicitní hodnotou 0.5.

iwin=18 **Exponenciální okno** (Exponential Window)

$$w(x) = e^{-\frac{|x|}{\tau}}, \quad -1 \leq x \leq 1,$$

kde τ je parametr okna (nastavený v **rwnd**).

iwin=19 **Welchovo okno** (Welch Window)

$$w(x) = 1 - x^2, \quad -1 \leq x \leq 1.$$

iwin=20 **Externě definované okno**

Pokud není uživatel spokojen s žádným s předdefinovaných oken, může si zadat své vlastní okno pomocí **N** vzorků ve vektoru odkazovaném vstupem **uwnd** předtím, než je spuštěn blok **PSD**.

Vstup **HLD** umožňuje dočasně zastavit výpočet **PSD**.

Tento blok nepropaguje kvalitu signálu. Více informací je uvedeno v sekci 1.4.

Vstup

uc	Vstupní odkaz na vstupní data	Reference
uf	Vstupní odkaz na interní vektor pro FFT	Reference
upsd	Vstupní odkaz na výstupní vektor/matice spektrální výkonové hustoty	Reference
urem	Vstupní odkaz na zbývající (nezpracovaná) data	Reference
uwnd	Vstupní odkaz na vektor váhové (window) funkce	Reference
uwrk	Vstupní odkaz na pracovní vektor/matice	Reference
n	Počet vzorků signálu v segmentu FFT	Long (I32)
fs	Frekvence vzorkování v [Hz]	Double (F64)
START	Start výpočtu (náběžná hrana)	Bool
HLD	Pozastavení	Bool

Parametr

mode	Výpočetní mód spektrální výkonové hustoty	⊙1	Long (I32)
	1 Reálná		
	2 Reálná s překryvem		
	3 Komplexní		
	4 Komplexní s překryvem		
mtrig	Spouštěcí mód výpočtu	⊙1	Long (I32)
	1 Každá perioda		
	2 Dostatek vzorků		
	3 Náběžná hrana START		

iwnd	Váhová funkce (window)	⊙1	Long (I32)
	1 Bartlett-Hann		
	2 Bartlett		
	3 Blackman		
	4 Blackman-Harris		
	5 Bohman		
	6 Čebyšev		
	7 Flat top		
	8 Gaussian		
	9 Hamming		
	10 Hann		
	11 —		
	12 Nuttall		
	13 Parzen		
	14 Obdélníkové		
	15 —		
	16 Trojúhelníkové		
	17 Tukey		
	18 Exponenciální		
	19 Welch		
	20 Externí		
lwnd	Parametr lwnd některých váhových funkcí	↓1 ↑9 ⊙1	Long (I32)
rwnd	Parametr rwnd některých váhových funkcí	↓1 ↑9 ⊙1	Long (I32)
navg	Počet segmentů FFT pro průměrování	↓1 ⊙10	Long (I32)
DB	Vypočtená spektrální výkonová hustota je konvertována do decibelů [dB]		Bool

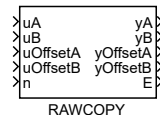
Výstup

yc	Výstupní odkaz na vstupní data	Reference
yf	Výstupní odkaz na interní vektor pro FFT	Reference
ypsd	Výstupní odkaz na výstupní vektor/matici spektrální výkonové hustoty	Reference
yrem	Výstupní odkaz na zbývající (nezpracovaná) data	Reference
ywnd	Výstupní odkaz na externí váhovou (window) funkci	Reference
ywrk	Výstupní odkaz na pracovní vektor/matici	Reference
E	Příznak chyby	Bool
DONE	Příznak dokončení výpočtu průměrováním	Bool
iavg	Aktuální index segmentu pro průměrování	Long (I32)
lcomp	Počet úspěšných výpočtů spektrální výkonové hustoty	Large (I64)

RAWCOPY – Kopírování vektoru po 1, 2 nebo 4 Bajtech

Symbol bloku

Licence: [ADVANCED](#)



Popis funkce

Popis tohoto bloku ještě není k dispozici. Níže naleznete částečný popis vstupů, výstupů a parametrů bloku. Kompletní popis bloku bude k dispozici v dalších revizích dokumentace.

Tento blok nepropaguje kvalitu signálu. Více informací je uvedeno v sekci [1.4](#).

Vstup

uA	Zdrojový vektor	Reference
uB	Cílový vektor	Reference
uOffsetA	Offset do zdrojového vektoru v bajtech	DWord (U32)
uOffsetB	Offset do cílového vektoru v bajtech	DWord (U32)
n	Počet bajtů ke zkopírování	DWord (U32)

Parametr

wrapA	Cyklování zdrojové adresy po x Bajtech (0 pro bez cyklování)	DWord (U32)
wrapB	Cyklování cílové adresy po x Bajtech (0 pro bez cyklování)	DWord (U32)
bytes	Počet bajtů ke zkopírování	⊙XXX DWord (U32)
	1 1 byte	
	2 2 bytes	
	4 4 bytes	
	8 8 bytes	

Výstup

yA	Výstupní odkaz na zdrojový vektor	Reference
yB	Výstupní odkaz na cílový vektor	Reference
yOffsetA	Offset do zdrojového vektoru po kopírování (v bajtech)	DWord (U32)
yOffsetB	Offset do cílového vektoru po kopírování (v bajtech)	DWord (U32)
E	Příznak chyby	Bool

Kapitola 19

MQTTDrv – Komunikace přes MQTT protokol

Obsah

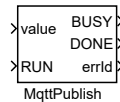
MqttPublish – Odeslání zprávy protokolem MQTT	652
MqttSubscribe – Odběr zpráv z MQTT topic	654

Knihovna MQTTDrv je určena pro komunikaci IoT (Internet of Things) pomocí protokolu MQTT (Message Queuing Telemetry Transport). Skládá se ze dvou primárních bloků: [MqttPublish](#) a [MqttSubscribe](#). Blok [MqttPublish](#) se používá pro odesílání zpráv brokerovi MQTT, umožňující publikování dat do MQTT témat. Naopak blok [MqttSubscribe](#) je určen pro přihlášení k odběru témat a příjem zpráv od brokera. Tato knihovna usnadňuje efektivní a efektivní datovou komunikaci v aplikacích IoT a využívá lehký a široce používaný protokol MQTT pro výměnu zpráv.

MqttPublish – Odeslání zprávy protokolem MQTT

Symbol bloku

Licence: [MQTT](#)



Popis funkce

Tento funkční blok je závislý na MQTT ovladači. Je doporučeno si před použitím přečíst manuál MQTTDrv ovladače [25].

Blok MqttPublish slouží pro odesílání zpráv na zařízení typu *MQTT Broker* prostřednictvím připojení, které zajišťuje MQTTDrv ovladač.

Prvním parametrem jménem *topic* se určuje do jakého tématického celku budou zprávy produkované blokem zařazeny. Protokol MQTT doručuje aplikační zprávy dle zvolené úrovně kvality služby (*Quality of Service – QoS*). Požadovaná úroveň lze nastavit parametrem *QoS*. Více informací naleznete ve specifikaci MQTT protokolu [26] (pouze anglicky).

Pokud je parametr **RETAIN** nahozen, odchozí pakety zprávy budou označeny příznakem **RETAIN**. Více informací naleznete ve specifikaci MQTT protokolu [26] (pouze anglicky).

Parametr **defBuffSize** může být použit pro optimalizaci práce s pamětí bloku. Hodnota parametru představuje velikost staticky alokované paměti pro vnitřní buffer odchozích zpráv. Pokud je velikost bufferu nadbytečně velká, blok si alokuje paměť, která je zbytečně blokována. Na druhou stranu pokud je hodnota parametru příliš malá, algoritmus bloku musí často dynamicky alokovat paměť, což je časově náročná operace.

Odesílaná zpráva je konstruována ze vstupního signálu **value**. Blok předpokládá, že vstup **value** bude datového typu *string*. Pokud je vstup jiného typu, dojde k automatické konverzi. Pro odeslání zprávy v aktuální periodě stačí nastavit vstup **RUN** na **on**. Výstup **BUSY** má hodnotu **on** pokud je blok zaneprázdněn nevyřízeným požadavkem pro odeslání zprávy nebo vyčkává na odpověď od zařízení typu *Broker*. Pokud je požadavek pro odeslání úspěšně vyřízen v aktuální periodě, dojde k nastavení výstupu **DONE** na **on**.

Tento blok nepropaguje kvalitu signálu. Více informací je uvedeno v sekci 1.4.

Vstup

value	Vstupní signál	String
RUN	Povolení běhu algoritmu	Bool

Parametr

<code>topic</code>	MQTT topic		<code>String</code>
<code>QoS</code>	Quality of Service	☉1	<code>Long (I32)</code>
	1 QoS0 (Maximálně jednou)		
	2 QoS1 (Alespoň jednou)		
	3 QoS2 (Právě jednou)		
<code>RETAIN</code>	Zachovat poslední zprávu	☉on	<code>Bool</code>
<code>defBuffSize</code>	Výchozí velikost bufferu	↓1 ☉2048	<code>Long (I32)</code>

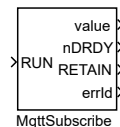
Výstup

<code>BUSY</code>	Příznak probíhající operace	<code>Bool</code>
<code>DONE</code>	Příznak dokončení transakce	<code>Bool</code>
<code>errId</code>	Kód chyby	<code>Error</code>

MqttSubscribe – Odběr zpráv z MQTT topic

Symbol bloku

Licence: [MQTT](#)



Popis funkce

Tento funkční blok je závislý na MQTT ovladači. Je doporučeno si před použitím přečíst manuál `MQTTRV` ovladače [25].

Účelem bloku `MqttSubscribe` je přihlášení se k odebírání zpráv ze zařízení typu *MQTT Broker* a jejich vyčítání prostřednictvím spojení, které zajišťuje ovladač `MQTTRV`.

Parametr `topic` určuje tématický celek, k jehož odběru se blok přihlašuje. Protokol MQTT doručuje aplikační zprávy dle zvolené úrovně kvality služby (*Quality of Service* – *QoS*). Požadovaná úroveň lze nastavit parametrem `QoS`. Více informací naleznete ve specifikaci MQTT protokolu [26] (pouze anglicky).

Nastavením parametru `type` lze určit, jaký je očekávaný typ čtených zpráv. Blok se snaží konvertovat příchozí zprávy do zvoleného datového typu a výsledek nastavuje na výstupní signál `value` v případě úspěchu a nebo nastavuje výstup `errId` na příslušný chybový kód.

Parametr `mode` má dvě možné hodnoty: `Last value` a `Buffered values`. Pokud je parametr nastaven na hodnotu `Last value`, blok na výstup vždy vystaví pouze poslední zprávu i v případě, že bylo v poslední periodě přijato více zpráv. Pokud je ale parametr nastaven na hodnotu `Buffered values`, blok si zprávy ukládá do vnitřního bufferu a na výstup předává jednu zprávu za druhou v následujících ticích tasku.

Parametr `defBuffSize` může být použit pro optimalizaci práce s pamětí bloku. Hodnota parametru představuje velikost staticky alokované paměti pro vnitřní buffer odchozích zpráv. Pokud je velikost bufferu nadbytečně velká, blok si alokuje paměť, která je zbytečně blokována. Na druhou stranu pokud je hodnota parametru příliš malá, algoritmus bloku musí často dynamicky alokovat paměť, což je časově náročná operace.

Akce přihlášení se k odběru je provedena na základě vzestupné hrany (`off`→`on`) a odhlášení odběru na základě sestupné hrany (`on`→`off`) vstupu `RUN`.

Výstup `nDRDY` určuje počet přijatých zpráv, které jsou dostupné ve vnitřním bufferu. Pokud je parametr `mode` nastaven na hodnotu `Last value` výstup může mít pouze hodnotu 0 nebo 1.

Výstup `RETAIN` je nastaven na `on`, pokud přijatý paket měl nastaven příznak `RETAIN`. Více informací naleznete ve specifikaci MQTT protokolu [26] (pouze anglicky)..

Přihlašování se k odběrům s využitím zástupných znaků typu *wildcards* není podporováno.

Tento blok nepropaguje kvalitu signálu. Více informací je uvedeno v sekci [1.4](#).

Vstup

RUN	Povolení běhu algoritmu	Bool

Parametr

topic	MQTT topic	String
QoS	Quality of Service	⊙1 Long (I32)
	1 QoS0 (Maximálně jednou)	
	2 QoS1 (Alespoň jednou)	
	3 QoS2 (Právě jednou)	
type	Očekávaný typ příchozích dat	⊙1 Long (I32)
	1 string	
	2 double	
	3 long	
	4 bool	
	5 byte vector/blob	
mode	Způsob bufferování příchozích zpráv	⊙1 Long (I32)
	1 poslední hodnota	
	2 buffrovat hodnoty	
defBuffSize	Výchozí velikost bufferu	↓1 ⊙2048 Long (I32)

Výstup

value	Výstupní signál	Any
nDRDY	Počet přijatých zpráv	↓0 ↑10 Long (I32)
RETAIN	Zachovat poslední zprávu	⊙on Bool
errId	Kód chyby	Error

Kapitola 20

MC_SINGLE – Řízení pohybu v jedné ose

Obsah

MCP_AccelerationProfile – * Generování trajektorie (zrychlení) .	660
MCP_Halt – * Zastavení pohybu (přerušitelné)	662
MCP_HaltSuperimposed – * Zastavení pohybu (přídavné a přerušitelné)	663
MCP_Home – * Nalezení výchozí polohy	664
MCP_MoveAbsolute – * Pohyb do pozice (absolutní souřadnice) . .	666
MCP_MoveAdditive – * Pohyb do pozice (relativně ke konci předchozího pohybu)	668
MCP_MoveContinuousAbsolute – * Pohyb do pozice (absolutní souřadnice)	670
MCP_MoveContinuousRelative – * Pohyb do pozice (relativně ke konci předchozího pohybu)	672
MCP_MoveRelative – * Kartézské transformace	674
MCP_MoveSuperimposed – * Pohyb do pozice (přídavný pohyb) . .	676
MCP_MoveVelocity – * Pohyb konstantní rychlostí	677
MCP_PositionProfile – * Generování trajektorie (poloha)	679
MCP_SetOverride – * Nastavení násobivých faktorů na ose	681
MCP_Stop – * Zastavení pohybu	682
MCP_TorqueControl – * Řízení síly/momentu	683
MCP_VelocityProfile – * Generování trajektorie (rychlost)	685
MC_AccelerationProfile, MCP_AccelerationProfile – Generování trajektorie (zrychlení)	687
MC_Halt, MCP_Halt – Zastavení pohybu (přerušitelné)	691
MC_HaltSuperimposed, MCP_HaltSuperimposed – Zastavení pohybu (přídavné a přerušitelné)	693

MC_Home, MCP_Home – Nalezení výchozí polohy	694
MC_MoveAbsolute, MCP_MoveAbsolute – Pohyb do pozice (absolutní souřadnice)	696
MC_MoveAdditive, MCP_MoveAdditive – Pohyb do pozice (relativně ke konci předchozího pohybu)	699
MC_MoveContinuousAbsolute, MCP_MoveContinuousAbsolute – Pohyb do pozice (absolutní souřadnice)	702
MC_MoveContinuousRelative, MCP_MoveContinuousRelative – Pohyb do pozice (relativně ke konci předchozího pohybu)	706
MC_MoveRelative, MCP_MoveRelative – Pohyb do pozice (relativně k okamžiku spuštění)	710
MC_MoveSuperimposed, MCP_MoveSuperimposed – Pohyb do pozice (přídavný pohyb)	713
MC_MoveVelocity, MCP_MoveVelocity – Pohyb konstantní rychlostí	716
MC_PositionProfile, MCP_PositionProfile – Generování trajektorie (poloha)	719
MC_Power – Aktivace osy	723
MC_ReadActualPosition – Skutečná poloha osy	724
MC_ReadAxisError – Chyba osy	725
MC_ReadBoolParameter – Čtení parametru (logická hodnota) . . .	726
MC_ReadParameter – Čtení parametru (číselná hodnota)	727
MC_ReadStatus – Stav osy	729
MC_Reset – Nulování chyb osy	731
MC_SetOverride, MCP_SetOverride – Nastavení násobivých faktorů na ose	732
MC_Stop, MCP_Stop – Zastavení pohybu	734
MC_TorqueControl, MCP_TorqueControl – Řízení síly/momentu . . .	736
MC_VelocityProfile, MCP_VelocityProfile – Generování trajektorie (rychlost)	739
MC_WriteBoolParameter – Nastavení parametru (logická hodnota)	743
MC_WriteParameter – Nastavení parametru (číselná hodnota) . . .	744
RM_Axis – Osa pro řízení pohybu	745
RM_AxisOut – Výstupní blok osy	752
RM_AxisSpline – Interpolace požadované polohy (rychlosti, zrychlení)	753
RM_HomeOffset – * Nastavení počáteční polohy pomocí offsetu . .	758
RM_Track – Sledování a krokování	759

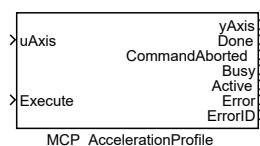
Knihovna MC_SINGLE je určena pro řízení pohybu v jednoosých systémech. Obsahuje bloky jako MC_MoveAbsolute, MC_MoveRelative a MC_MoveVelocity pro přesné polohování a ovládání rychlosti. Knihovna obsahuje MC_Home pro operace navádění a MC_Power pro řízení stavu napájení osy. Pokročilé funkce poskytují MC_AccelerationProfile,

`MC_PositionProfile` a `MC_VelocityProfile` pro přizpůsobení profilů pohybu. Nabízí také možnosti monitorování a úpravy parametrů prostřednictvím `MC_ReadActualPosition`, `MC_ReadAxisError`, `MC_ReadParameter` a `MC_WriteParameter`. Knihovna navíc obsahuje bloky jako `MC_Halt`, `MC_Reset` a `MC_Stop` pro nouzové a kontrolní operace. Tato knihovna je nezbytná pro aplikace vyžadující přesný a řízený pohyb v jednoosých konfiguracích.

MCP_AccelerationProfile – * Generování trajektorie (zrychlení)

Symbol bloku

Licence: [MOTION CONTROL](#)



Popis funkce

Popis tohoto bloku ještě není k dispozici. Níže naleznete částečný popis vstupů, výstupů a parametrů bloku. Kompletní popis bloku bude k dispozici v dalších revizích dokumentace.

Tento blok nepropaguje kvalitu signálu. Více informací je uvedeno v sekci [1.4](#).

Vstup

uAxis	Odkaz na osu (přípustné je jen spojení RM_Axis.axisRef-uAxis nebo yAxis-uAxis)	Reference
Execute	Náběžná hrana aktivuje blok	Bool

Parametr

alg	Typ interpolace	⊙1	Long (I32)
	1 tabulka čas/hodnota		
	2 hodnoty ve stejném intervalu		
	3 aproximace polynomy		
	4 polynomy s ekvidistantními intervaly		
	5 tabulka čas/hodnota (+okraj)		
	6 hodnoty ve stejném intervalu (+okraj)		
	7 aproximace polynomem 3.řádu (B-spline)		
	8 aproximace polynomem 5.řádu (B-spline)		
	9 vše lineární		
nmax	Počet segmentů profilu	⊙3	Long (I32)
TimeScale	Konstanta násobení pro přepočet časové osy profilu	⊙1.0	Double (F64)
AccelerationScale	Konstanta násobení pro přepočet hodnotové osy profilu	⊙1.0	Double (F64)
Offset	Aditivní konstanta pro přepočet hodnotové osy profilu		Double (F64)

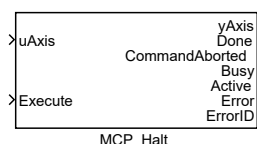
BufferMode	Režim převzetí osy	⊙1	Long (I32)
	1 aborting		
	2 buffered		
	3 blending low		
	4 blending high		
	5 blending previous		
	6 blending next		
times	Posloupnost hraničních časů jednotlivých segmentů	⊙[0 30]	Double (F64)
values	Hodnoty veličiny nebo koeficienty interpolačních polynomů (a0, a1, a2, ...)	⊙[0 100 100 50]	Double (F64)

Výstup

yAxis	Odkaz na osu (přípustné je jen spojení RM_Axis.axisRef-uAxis nebo yAxis-uAxis)	Reference
Done	Příznak dokončení algoritmu	Bool
CommandAborted	Příznak přerušení funkce bloku	Bool
Busy	Příznak, že algoritmus ještě neskončil	Bool
Active	Příznak, že blok řídí osu	Bool
Error	Příznak chyby	Bool
ErrorID	Výsledek poslední operace	Error
	i kód chyby REXYGENu	

MCP_Halt – * Zastavení pohybu (přerušitelné)

Symbol bloku

Licence: [MOTION CONTROL](#)

Popis funkce

Popis tohoto bloku ještě není k dispozici. Níže naleznete částečný popis vstupů, výstupů a parametrů bloku. Kompletní popis bloku bude k dispozici v dalších revizích dokumentace.

Tento blok nepropaguje kvalitu signálu. Více informací je uvedeno v sekci [1.4](#).

Vstup

uAxis	Odkaz na osu (přípustné je jen spojení RM_Axis.axisRef-uAxis nebo yAxis-uAxis)	Reference
Execute	Náběžná hrana aktivuje blok	Bool

Parametr

Deceleration	Maximální povolené zpomalení [unit/s ²]	Double (F64)
Jerk	Maximální povolená změna zrychlení [unit/s ³]	Double (F64)
BufferMode	Režim převzetí osy	⊙1 Long (I32)
	1 aborting	
	2 buffered	

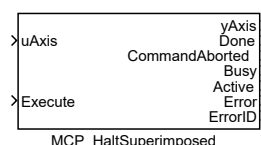
Výstup

yAxis	Odkaz na osu (přípustné je jen spojení RM_Axis.axisRef-uAxis nebo yAxis-uAxis)	Reference
Done	Příznak dokončení algoritmu	Bool
CommandAborted	Příznak přerušení funkce bloku	Bool
Busy	Příznak, že algoritmus ještě neskončil	Bool
Active	Příznak, že blok řídí osu	Bool
Error	Příznak chyby	Bool
ErrorID	Výsledek poslední operace	Error
	i kód chyby REXYGENu	

MCP_HaltSuperimposed – * Zastavení pohybu (přídavné a přerušitelné)

Symbol bloku

Licence: [MOTION CONTROL](#)



Popis funkce

Popis tohoto bloku ještě není k dispozici. Níže naleznete částečný popis vstupů, výstupů a parametrů bloku. Kompletní popis bloku bude k dispozici v dalších revizích dokumentace.

Tento blok nepropaguje kvalitu signálu. Více informací je uvedeno v sekci [1.4](#).

Vstup

uAxis	Odkaz na osu (přípustné je jen spojení RM_Axis.axisRef-uAxis nebo yAxis-uAxis)	Reference
Execute	Náběžná hrana aktivuje blok	Bool

Parametr

Deceleration	Maximální povolené zpomalení [unit/s ²]	Double (F64)
Jerk	Maximální povolená změna zrychlení [unit/s ³]	Double (F64)
BufferMode	Režim převzetí osy	⊙1 Long (I32)
	1 aborting	
	2 buffered	

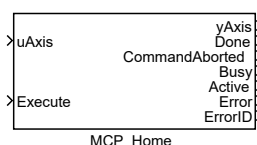
Výstup

yAxis	Odkaz na osu (přípustné je jen spojení RM_Axis.axisRef-uAxis nebo yAxis-uAxis)	Reference
Done	Příznak dokončení algoritmu	Bool
CommandAborted	Příznak přerušení funkce bloku	Bool
Busy	Příznak, že algoritmus ještě neskončil	Bool
Active	Příznak, že blok řídí osu	Bool
Error	Příznak chyby	Bool
ErrorID	Výsledek poslední operace	Error
	i kód chyby REXYGENu	

MCP_Home – * Nalezení výchozí polohy

Symbol bloku

Licence: MOTION CONTROL



Popis funkce

Popis tohoto bloku ještě není k dispozici. Níže naleznete částečný popis vstupů, výstupů a parametrů bloku. Kompletní popis bloku bude k dispozici v dalších revizích dokumentace.

Tento blok nepropaguje kvalitu signálu. Více informací je uvedeno v sekci 1.4.

Vstup

uAxis	Odkaz na osu (přípustné je jen spojení RM_Axis.axisRef-uAxis nebo yAxis-uAxis)	Reference
Execute	Náběžná hrana aktivuje blok	Bool

Parametr

Velocity	Maximální povolená rychlost [unit/s]	Double (F64)
Acceleration	Maximální povolené zrychlení [unit/s^2]	Double (F64)
TorqueLimit	Maximální povolený moment/síla	Double (F64)
TimeLimit	Maximální povolený čas pro celý algoritmus bloku [s]	Double (F64)
DistanceLimit	Maximální povolená vzdálenost pro celý algoritmus bloku [unit]	Double (F64)
LagLimit	Maximální povolený skluz pro celý algoritmus bloku [unit]	Double (F64)
Direction	Směr pohybu (jen pro cyklické osy nebo speciální případy) 1 kladný 2 nejkratší 3 záporný 4 aktuální	⊙3 Long (I32)
Position_	Požadovaná poloha [unit]	Double (F64)

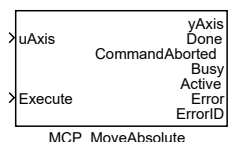
HomingMode	Algoritmus hledání výchozí pozice	⊙1	Long (I32)
1	 nulový spínač		
2	 koncové spínače		
3	 nulový pulz		
4	 uživatelská hodnota		
5	 absolutní snímač polohy		
6	 mechanický doraz		
7	 reserva1		
8	 reserva2		

Výstup

yAxis	Odkaz na osu (přípustné je jen spojení RM_Axis.axisRef-uAxis nebo yAxis-uAxis)	Reference
Done	Příznak dokončení algoritmu	Bool
CommandAborted	Příznak přerušení funkce bloku	Bool
Busy	Příznak, že algoritmus ještě neskončil	Bool
Active	Příznak, že blok řídí osu	Bool
Error	Příznak chyby	Bool
ErrorID	Výsledek poslední operace	Error
i	 kód chyby REXYGENu	

MCP_MoveAbsolute – * Pohyb do pozice (absolutní souřadnice)

Symbol bloku

Licence: [MOTION CONTROL](#)

Popis funkce

Popis tohoto bloku ještě není k dispozici. Níže naleznete částečný popis vstupů, výstupů a parametrů bloku. Kompletní popis bloku bude k dispozici v dalších revizích dokumentace.

Tento blok nepropaguje kvalitu signálu. Více informací je uvedeno v sekci [1.4](#).

Vstup

uAxis	Odkaz na osu (přípustné je jen spojení RM_Axis.axisRef-uAxis nebo yAxis-uAxis)	Reference
Execute	Náběžná hrana aktivuje blok	Bool

Parametr

Position_	Požadovaná poloha [unit]		Double (F64)
Velocity	Maximální povolená rychlost [unit/s]	↓0.0	Double (F64)
Acceleration	Maximální povolené zrychlení [unit/s^2]	↓0.0	Double (F64)
Deceleration	Maximální povolené zpomalení [unit/s^2]	↓0.0	Double (F64)
Jerk	Maximální povolená změna zrychlení [unit/s^3]	↓0.0	Double (F64)
BufferMode	Režim převzetí osy	⊙1	Long (I32)
	1 aborting		
	2 buffered		
	3 blending low		
	4 blending high		
	5 blending previous		
	6 blending next		
Direction	Směr pohybu (jen pro cyklické osy nebo speciální případy)	⊙1	Long (I32)
	1 kladný		
	2 nejkratší		
	3 záporný		
	4 aktuální		

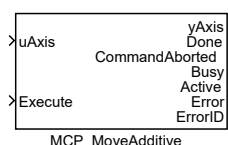
Výstup

<code>yAxis</code>	Odkaz na osu (přípustné je jen spojení <code>RM_Axis.axisRef-uAxis</code> nebo <code>yAxis-uAxis</code>)	<code>Reference</code>
<code>Done</code>	Příznak dokončení algoritmu	<code>Bool</code>
<code>CommandAborted</code>	Příznak přerušení funkce bloku	<code>Bool</code>
<code>Busy</code>	Příznak, že algoritmus ještě neskončil	<code>Bool</code>
<code>Active</code>	Příznak, že blok řídí osu	<code>Bool</code>
<code>Error</code>	Příznak chyby	<code>Bool</code>
<code>ErrorID</code>	Výsledek poslední operace i kód chyby <code>REXYGENu</code>	<code>Error</code>

MCP_MoveAdditive – * Pohyb do pozice (relativně ke konci předchozího pohybu)

Symbol bloku

Licence: [MOTION CONTROL](#)



Popis funkce

Popis tohoto bloku ještě není k dispozici. Níže naleznete částečný popis vstupů, výstupů a parametrů bloku. Kompletní popis bloku bude k dispozici v dalších revizích dokumentace.

Tento blok nepropaguje kvalitu signálu. Více informací je uvedeno v sekci [1.4](#).

Vstup

uAxis	Odkaz na osu (přípustné je jen spojení RM_Axis.axisRef-uAxis nebo yAxis-uAxis)	Reference
Execute	Náběžná hrana aktivuje blok	Bool

Parametr

Distance	Požadovaná vzdálenost (od okamžiku začátku řízení osy blokem) [unit]	Double (F64)
Velocity	Maximální povolená rychlost [unit/s]	↓0.0 Double (F64)
Acceleration	Maximální povolené zrychlení [unit/s^2]	↓0.0 Double (F64)
Deceleration	Maximální povolené zpomalení [unit/s^2]	↓0.0 Double (F64)
Jerk	Maximální povolená změna zrychlení [unit/s^3]	↓0.0 Double (F64)
BufferMode	Režim převzetí osy	⊙1 Long (I32)
	1 aborting	
	2 buffered	
	3 blending low	
	4 blending high	
	5 blending previous	
	6 blending next	

Výstup

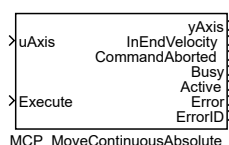
yAxis	Odkaz na osu (přípustné je jen spojení RM_Axis.axisRef-uAxis nebo yAxis-uAxis)	Reference
Done	Příznak dokončení algoritmu	Bool
CommandAborted	Příznak přerušení funkce bloku	Bool

Busy	Příznak, že algoritmus ještě neskončil	Bool
Active	Příznak, že blok řídí osu	Bool
Error	Příznak chyby	Bool
ErrorID	Výsledek poslední operace i kód chyby REXYGENu	Error

MCP_MoveContinuousAbsolute – * Pohyb do pozice (absolutní souřadnice)

Symbol bloku

Licence: [MOTION CONTROL](#)



Popis funkce

Popis tohoto bloku ještě není k dispozici. Níže naleznete částečný popis vstupů, výstupů a parametrů bloku. Kompletní popis bloku bude k dispozici v dalších revizích dokumentace.

Tento blok nepropaguje kvalitu signálu. Více informací je uvedeno v sekci [1.4](#).

Vstup

uAxis	Odkaz na osu (přípustné je jen spojení RM_Axis.axisRef-uAxis nebo yAxis-uAxis)	Reference
Execute	Náběžná hrana aktivuje blok	Bool

Parametr

Position_	Požadovaná poloha [unit]	Double (F64)
Velocity	Maximální povolená rychlost [unit/s]	↓0.0 Double (F64)
Acceleration	Maximální povolené zrychlení [unit/s^2]	↓0.0 Double (F64)
Deceleration	Maximální povolené zpomalení [unit/s^2]	↓0.0 Double (F64)
Jerk	Maximální povolená změna zrychlení [unit/s^3]	↓0.0 Double (F64)
BufferMode	Režim převzetí osy	⊙1 Long (I32)
	1 aborting	
	2 buffered	
	3 blending low	
	4 blending high	
	5 blending previous	
	6 blending next	
Direction	Směr pohybu (jen pro cyklické osy nebo speciální případy)	⊙1 Long (I32)
	1 kladný	
	2 nejkratší	
	3 záporný	
	4 aktuální	
EndVelocity	Koncová rychlost	Double (F64)

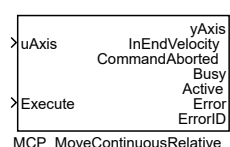
Výstup

<code>yAxis</code>	Odkaz na osu (přípustné je jen spojení <code>RM_Axis.axisRef-uAxis</code> nebo <code>yAxis-uAxis</code>)	<code>Reference</code>
<code>InEndVelocity</code>	Příznak dokončení algoritmu	<code>Bool</code>
<code>CommandAborted</code>	Příznak přerušení funkce bloku	<code>Bool</code>
<code>Busy</code>	Příznak, že algoritmus ještě neskončil	<code>Bool</code>
<code>Active</code>	Příznak, že blok řídí osu	<code>Bool</code>
<code>Error</code>	Příznak chyby	<code>Bool</code>
<code>ErrorID</code>	Výsledek poslední operace i kód chyby <code>REXYGENu</code>	<code>Error</code>

MCP_MoveContinuousRelative – * Pohyb do pozice (relativně ke konci předchozího pohybu)

Symbol bloku

Licence: [MOTION CONTROL](#)



Popis funkce

Popis tohoto bloku ještě není k dispozici. Níže naleznete částečný popis vstupů, výstupů a parametrů bloku. Kompletní popis bloku bude k dispozici v dalších revizích dokumentace.

Tento blok nepropaguje kvalitu signálu. Více informací je uvedeno v sekci [1.4](#).

Vstup

uAxis	Odkaz na osu (přípustné je jen spojení RM_Axis.axisRef-uAxis nebo yAxis-uAxis)	Reference
Execute	Náběžná hrana aktivuje blok	Bool

Parametr

Distance	Požadovaná vzdálenost (od okamžiku startu bloku) [unit]	Double (F64)
Velocity	Maximální povolená rychlost [unit/s]	↓0.0 Double (F64)
Acceleration	Maximální povolené zrychlení [unit/s^2]	↓0.0 Double (F64)
Deceleration	Maximální povolené zpomalení [unit/s^2]	↓0.0 Double (F64)
Jerk	Maximální povolená změna zrychlení [unit/s^3]	↓0.0 Double (F64)
BufferMode	Režim převzetí osy	⊙1 Long (I32)
	1 aborting	
	2 buffered	
	3 blending low	
	4 blending high	
	5 blending previous	
	6 blending next	
EndVelocity	Koncová rychlost	Double (F64)

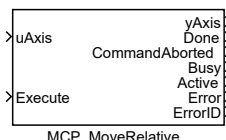
Výstup

yAxis	Odkaz na osu (přípustné je jen spojení RM_Axis.axisRef-uAxis nebo yAxis-uAxis)	Reference
InEndVelocity	Příznak dokončení algoritmu	Bool
CommandAborted	Příznak přerušení funkce bloku	Bool

Busy	Příznak, že algoritmus ještě neskončil	Bool
Active	Příznak, že blok řídí osu	Bool
Error	Příznak chyby	Bool
ErrorID	Výsledek poslední operace i kód chyby REXYGENu	Error

MCP_MoveRelative – * **Kartézské transformace**

Symbol bloku

Licence: **MOTION CONTROL**

Popis funkce

Popis tohoto bloku ještě není k dispozici. Níže naleznete částečný popis vstupů, výstupů a parametrů bloku. Kompletní popis bloku bude k dispozici v dalších revizích dokumentace.

Tento blok nepropaguje kvalitu signálu. Více informací je uvedeno v sekci [1.4](#).

Vstup

uAxis	Odkaz na osu (přípustné je jen spojení RM_Axis.axisRef-uAxis nebo yAxis-uAxis)	Reference
Execute	Náběžná hrana aktivuje blok	Bool

Parametr

Distance	Požadovaná vzdálenost (od okamžiku startu bloku) [unit]	Double (F64)
Velocity	Maximální povolená rychlost [unit/s]	↓0.0 Double (F64)
Acceleration	Maximální povolené zrychlení [unit/s^2]	↓0.0 Double (F64)
Deceleration	Maximální povolené zpomalení [unit/s^2]	↓0.0 Double (F64)
Jerk	Maximální povolená změna zrychlení [unit/s^3]	↓0.0 Double (F64)
BufferMode	Režim převzetí osy	⊙1 Long (I32)
	1 aborting	
	2 buffered	
	3 blending low	
	4 blending high	
	5 blending previous	
	6 blending next	

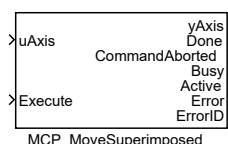
Výstup

yAxis	Odkaz na osu (přípustné je jen spojení RM_Axis.axisRef-uAxis nebo yAxis-uAxis)	Reference
Done	Příznak dokončení algoritmu	Bool
CommandAborted	Příznak přerušení funkce bloku	Bool
Busy	Příznak, že algoritmus ještě neskončil	Bool
Active	Příznak, že blok řídí osu	Bool

Error	Příznak chyby	Bool
ErrorID	Výsledek poslední operace	Error
	i kód chyby REXYGENu	

MCP_MoveSuperimposed – * Pohyb do pozice (přídavný pohyb)

Symbol bloku

Licence: [MOTION CONTROL](#)

Popis funkce

Popis tohoto bloku ještě není k dispozici. Níže naleznete částečný popis vstupů, výstupů a parametrů bloku. Kompletní popis bloku bude k dispozici v dalších revizích dokumentace.

Tento blok nepropaguje kvalitu signálu. Více informací je uvedeno v sekci [1.4](#).

Vstup

uAxis	Odkaz na osu (přípustné je jen spojení RM_Axis.axisRef-uAxis nebo yAxis-uAxis)	Reference
Execute	Náběžná hrana aktivuje blok	Bool

Parametr

Distance	Požadovaná vzdálenost (od okamžiku startu bloku) [unit]	Double (F64)
VelocityDiff	Maximální povolená rychlost [unit/s]	↓0.0 Double (F64)
Acceleration	Maximální povolené zrychlení [unit/s^2]	↓0.0 Double (F64)
Deceleration	Maximální povolené zpomalení [unit/s^2]	↓0.0 Double (F64)
Jerk	Maximální povolená změna zrychlení [unit/s^3]	↓0.0 Double (F64)

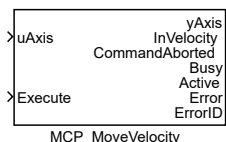
Výstup

yAxis	Odkaz na osu (přípustné je jen spojení RM_Axis.axisRef-uAxis nebo yAxis-uAxis)	Reference
Done	Příznak dokončení algoritmu	Bool
CommandAborted	Příznak přerušení funkce bloku	Bool
Busy	Příznak, že algoritmus ještě neskončil	Bool
Active	Příznak, že blok řídí osu	Bool
Error	Příznak chyby	Bool
ErrorID	Výsledek poslední operace i kód chyby REXYGENu	Error

MCP_MoveVelocity – * Pohyb konstantní rychlostí

Symbol bloku

Licence: [MOTION CONTROL](#)



Popis funkce

Popis tohoto bloku ještě není k dispozici. Níže naleznete částečný popis vstupů, výstupů a parametrů bloku. Kompletní popis bloku bude k dispozici v dalších revizích dokumentace.

Tento blok nepropaguje kvalitu signálu. Více informací je uvedeno v sekci [1.4](#).

Vstup

uAxis	Odkaz na osu (přípustné je jen spojení RM_Axis.axisRef-uAxis nebo yAxis-uAxis)	Reference
Execute	Náběžná hrana aktivuje blok	Bool

Parametr

Velocity	Maximální povolená rychlost [unit/s]	Double (F64)
Acceleration	Maximální povolené zrychlení [unit/s ²]	Double (F64)
Deceleration	Maximální povolené zpomalení [unit/s ²]	Double (F64)
Jerk	Maximální povolená změna zrychlení [unit/s ³]	Double (F64)
Direction	Směr pohybu (jen pro cyklické osy nebo speciální případy) $\odot 1$	Long (I32)
	1 kladný	
	2 nejkratší	
	3 záporný	
	4 aktuální	
BufferMode	Režim převzetí osy $\odot 1$	Long (I32)
	1 aborting	
	2 buffered	
	3 blending low	
	4 blending high	
	5 blending previous	
	6 blending next	

Výstup

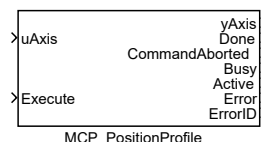
yAxis	Odkaz na osu (přípustné je jen spojení RM_Axis.axisRef-uAxis nebo yAxis-uAxis)	Reference

InVelocity	Příznak dosažení požadované rychlosti	Bool
CommandAborted	Příznak přerušení funkce bloku	Bool
Busy	Příznak, že algoritmus ještě neskončil	Bool
Active	Příznak, že blok řídí osu	Bool
Error	Příznak chyby	Bool
ErrorID	Výsledek poslední operace i kód chyby REXYGENu	Error

MCP_PositionProfile – * Generování trajektorie (poloha)

Symbol bloku

Licence: [MOTION CONTROL](#)



Popis funkce

Popis tohoto bloku ještě není k dispozici. Níže naleznete částečný popis vstupů, výstupů a parametrů bloku. Kompletní popis bloku bude k dispozici v dalších revizích dokumentace.

Tento blok nepropaguje kvalitu signálu. Více informací je uvedeno v sekci [1.4](#).

Vstup

uAxis	Odkaz na osu (přípustné je jen spojení RM_Axis.axisRef-uAxis nebo yAxis-uAxis)	Reference
Execute	Náběžná hrana aktivuje blok	Bool

Parametr

alg	Typ interpolace	⊙2	Long (I32)
	1 tabulka čas/hodnota		
	2 hodnoty ve stejném intervalu		
	3 aproximace polynomy		
	4 polynomy s ekvidistantními intervaly		
	5 tabulka čas/hodnota (+okraj)		
	6 hodnoty ve stejném intervalu (+okraj)		
	7 aproximace polynomem 3.řádu (B-spline)		
	8 aproximace polynomem 5.řádu (B-spline)		
	9 vše lineární		
nmax	Počet segmentů profilu	⊙3	Long (I32)
TimeScale	Konstanta násobení pro přepočít časové osy profilu	⊙1.0	Double (F64)
PositionScale	Konstanta násobení pro přepočít hodnotové osy profilu	⊙1.0	Double (F64)
Offset	Aditivní konstanta pro přepočít hodnotové osy profilu		Double (F64)
BufferMode	Režim převzetí osy	⊙1	Long (I32)
	1 aborting		
	2 buffered		
	3 blending low		
	4 blending high		
	5 blending previous		
	6 blending next		

<code>BeginVelocity</code>	rychlost na počátku pohybu	Double (F64)
<code>EndVelocity</code>	rychlost na konci pohybu	Double (F64)
<code>times</code>	Posloupnost hraničních časů jednotlivých segmentů $\odot[0\ 30]$	Double (F64)
<code>values</code>	Hodnoty veličiny nebo koeficienty interpolačních polynomů (a_0 , a_1 , a_2 , ...) $\odot[0\ 100\ 100\ 50]$	Double (F64)

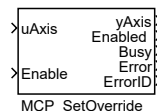
Výstup

<code>yAxis</code>	Odkaz na osu (přípustné je jen spojení <code>RM_Axis.axisRef-uAxis</code> nebo <code>yAxis-uAxis</code>)	Reference
<code>Done</code>	Příznak dokončení algoritmu	Bool
<code>CommandAborted</code>	Příznak přerušení funkce bloku	Bool
<code>Busy</code>	Příznak, že algoritmus ještě neskončil	Bool
<code>Active</code>	Příznak, že blok řídí osu	Bool
<code>Error</code>	Příznak chyby	Bool
<code>ErrorID</code>	Výsledek poslední operace i kód chyby <code>REXYGENu</code>	Error

MCP_SetOverride — * Nastavení násobivých faktorů na ose

Symbol bloku

Licence: [MOTION CONTROL](#)



Popis funkce

Popis tohoto bloku ještě není k dispozici. Níže naleznete částečný popis vstupů, výstupů a parametrů bloku. Kompletní popis bloku bude k dispozici v dalších revizích dokumentace.

Tento blok nepropaguje kvalitu signálu. Více informací je uvedeno v sekci [1.4](#).

Vstup

uAxis	Odkaz na osu (přípustné je jen spojení RM_Axis.axisRef-uAxis nebo yAxis-uAxis)	Reference
Enable	Povolení funkce bloku (aktivace výstupů)	Bool

Parametr

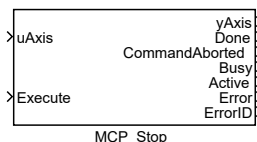
diff	Minimální hodnota změny násobivého faktoru	↓0.0 ↑1.0	Double (F64)
VelFactor	Faktor násobení pro rychlost	⊙1.0	Double (F64)
AccFactor	Faktor násobení pro zrychlení	⊙1.0	Double (F64)
JerkFactor	Faktor násobení pro změnu zrychlení	⊙1.0	Double (F64)

Výstup

yAxis	Odkaz na osu (přípustné je jen spojení RM_Axis.axisRef-uAxis nebo yAxis-uAxis)	Reference
Enabled	Povolení funkce bloku (aktivace výstupů)	Bool
Busy	Příznak, že algoritmus ještě neskončil	Bool
Error	Příznak chyby	Bool
ErrorID	Výsledek poslední operace i kód chyby REXYGENu	Error

MCP_Stop – * Zastavení pohybu

Symbol bloku

Licence: [MOTION CONTROL](#)

Popis funkce

Popis tohoto bloku ještě není k dispozici. Níže naleznete částečný popis vstupů, výstupů a parametrů bloku. Kompletní popis bloku bude k dispozici v dalších revizích dokumentace.

Tento blok nepropaguje kvalitu signálu. Více informací je uvedeno v sekci [1.4](#).

Vstup

uAxis	Odkaz na osu (přípustné je jen spojení RM_Axis.axisRef-uAxis nebo yAxis-uAxis)	Reference
Execute	Náběžná hrana aktivuje blok	Bool

Parametr

Deceleration	Maximální povolené zpomalení [unit/s ²]	Double (F64)
Jerk	Maximální povolená změna zrychlení [unit/s ³]	Double (F64)

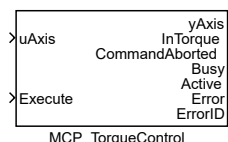
Výstup

yAxis	Odkaz na osu (přípustné je jen spojení RM_Axis.axisRef-uAxis nebo yAxis-uAxis)	Reference
Done	Příznak dokončení algoritmu	Bool
CommandAborted	Příznak přerušení funkce bloku	Bool
Busy	Příznak, že algoritmus ještě neskončil	Bool
Active	Příznak, že blok řídí osu	Bool
Error	Příznak chyby	Bool
ErrorID	Výsledek poslední operace i kód chyby REXYGENu	Error

MCP_TorqueControl – * Řízení síly/momentu

Symbol bloku

Licence: [MOTION CONTROL](#)



Popis funkce

Popis tohoto bloku ještě není k dispozici. Níže naleznete částečný popis vstupů, výstupů a parametrů bloku. Kompletní popis bloku bude k dispozici v dalších revizích dokumentace.

Tento blok nepropaguje kvalitu signálu. Více informací je uvedeno v sekci [1.4](#).

Vstup

uAxis	Odkaz na osu (přípustné je jen spojení RM_Axis.axisRef-uAxis nebo yAxis-uAxis)	Reference
Execute	Náběžná hrana aktivuje blok	Bool

Parametr

kma	Poměr mezi silou/momentem a zrychlením	Double (F64)
Torque	Maximální povolený moment/síla	Double (F64)
TorqueRamp	Maximální povolená změna momentu/síly	Double (F64)
Velocity	Maximální povolená rychlost [unit/s]	Double (F64)
Acceleration	Maximální povolené zrychlení [unit/s^2]	Double (F64)
Deceleration	Maximální povolené zpomalení [unit/s^2]	Double (F64)
Jerk	Maximální povolená změna zrychlení [unit/s^3]	Double (F64)
Direction	Směr pohybu (jen pro cyklické osy nebo speciální případy)	⊙1 Long (I32)
	1 kladný	
	2 nejkratší	
	3 záporný	
	4 aktuální	
BufferMode	Režim převzetí osy	⊙1 Long (I32)
	1 aborting	
	2 buffered	
	3 blending low	
	4 blending high	
	5 blending previous	
	6 blending next	

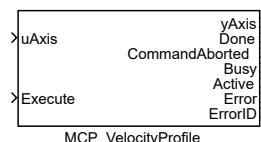
Výstup

yAxis	Odkaz na osu (přípustné je jen spojení RM_Axis.axisRef-uAxis nebo yAxis-uAxis)	Reference
InTorque	Příznak dosažení požadovaného momentu/síly	Bool
CommandAborted	Příznak přerušení funkce bloku	Bool
Busy	Příznak, že algoritmus ještě neskončil	Bool
Active	Příznak, že blok řídí osu	Bool
Error	Příznak chyby	Bool
ErrorID	Výsledek poslední operace i kód chyby REXYGENu	Error

MCP_VelocityProfile – * Generování trajektorie (rychlost)

Symbol bloku

Licence: [MOTION CONTROL](#)



Popis funkce

Popis tohoto bloku ještě není k dispozici. Níže naleznete částečný popis vstupů, výstupů a parametrů bloku. Kompletní popis bloku bude k dispozici v dalších revizích dokumentace.

Tento blok nepropaguje kvalitu signálu. Více informací je uvedeno v sekci [1.4](#).

Vstup

uAxis	Odkaz na osu (přípustné je jen spojení RM_Axis.axisRef-uAxis nebo yAxis-uAxis)	Reference
Execute	Náběžná hrana aktivuje blok	Bool

Parametr

alg	Typ interpolace	⊙1	Long (I32)
	1 tabulka čas/hodnota		
	2 hodnoty ve stejném intervalu		
	3 aproximace polynomy		
	4 polynomy s ekvidistantními intervaly		
	5 tabulka čas/hodnota (+okraj)		
	6 hodnoty ve stejném intervalu (+okraj)		
	7 aproximace polynomem 3.řádu (B-spline)		
	8 aproximace polynomem 5.řádu (B-spline)		
	9 vše lineární		
nmax	Počet segmentů profilu	⊙3	Long (I32)
TimeScale	Konstanta násobení pro přepočtení časové osy profilu	⊙1.0	Double (F64)
VelocityScale	Konstanta násobení pro přepočtení hodnotové osy profilu	⊙1.0	Double (F64)
Offset	Aditivní konstanta pro přepočtení hodnotové osy profilu		Double (F64)
BufferMode	Režim převzetí osy	⊙1	Long (I32)
	1 aborting		
	2 buffered		
	3 blending low		
	4 blending high		
	5 blending previous		
	6 blending next		

BeginAcceleration	zrychlení na počátku pohybu	Double (F64)
EndAcceleration	zrychlení na konci pohybu	Double (F64)
times	Posloupnost hraničních časů jednotlivých segmentů ⊙[0 15 25 30]	Double (F64)
values	Hodnoty veličiny nebo koeficienty interpolačních polynomů (a0, a1, a2, ...) ⊙[0 100 100 50]	Double (F64)

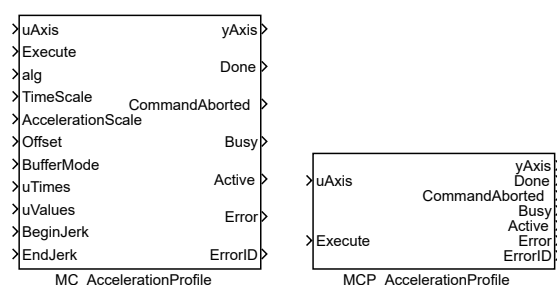
Výstup

yAxis	Odkaz na osu (přípustné je jen spojení RM_Axis.axisRef-uAxis nebo yAxis-uAxis)	Reference
Done	Příznak dokončení algoritmu	Bool
CommandAborted	Příznak přerušení funkce bloku	Bool
Busy	Příznak, že algoritmus ještě neskončil	Bool
Active	Příznak, že blok řídí osu	Bool
Error	Příznak chyby	Bool
ErrorID	Výsledek poslední operace i kód chyby REXYGENu	Error

MC_AccelerationProfile, MCP_AccelerationProfile – Generování trajektorie (zrychlení)

Symbody bloků

Licence: [MOTION CONTROL](#)



Popis funkce

Bloky MC_AccelerationProfile a MCP_AccelerationProfile mají naprosto shodnou funkci, jediným rozdílem je, že MCP_ varianta bloku má méně vstupů a potřebné konstanty se zadávají jako parametry bloku.

Popis funkce

Blok MC_AccelerationProfile generuje takovou trajektorii, aby zrychlení byla požadovaná funkce času. Existují dvě možnosti, jak tuto funkci zadat:

1. tabulkou: zadávají se dvojice čísel čas a zrychlení. Mezi jednotlivými časy se hodnota zrychlení interpoluje lineárně. Hodnoty času (v sekundách) se zadávají do pole/parametru **times**, příslušné hodnoty zrychlení do pole/parametru **values**. Posloupnost časových okamžiků musí být stoupající a musí začínat od 0 (resp. může začínat i zápornými hodnotami, ale profil se vykonává od času 0).

2. polynomy: celá funkce se v časové ose rozdělí na několik intervalů a pro každý interval se zadá aproximující polynom pátého řádu. Časové intervaly se definují jako v předchozím případě v poli **times**. Polynom pro každý interval je ve tvaru $p(x) = a_5x^5 + a_4x^4 + a_3x^3 + a_2x^2 + a_1x + a_0$, přičemž na začátku časového intervalu je $x = 0$, a na konci $x = 1$. Koeficienty a_i jsou uloženy v poli **values** ve vzestupném pořadí (tj. pole **values** obsahuje 6 hodnot pro každý časový interval). Tato metoda umožňuje snížit počet intervalů a pro určení koeficientů polynomů existuje speciální grafický editor.

Pro obě varianty je možné zvolit rozdělení na stejně dlouhé intervaly. pak je v poli **times** jen počáteční (obvykle 0) a koncový čas.

Poznámka 1: Vstup/Parametr **uValues** musí být ve všech případech vektor - nesmí to být matice, tj. jednotlivé hodnoty nesmí být odděleny středníkem (lze použít mezeru

nebo čárku).

Poznámka 2: V režimu zadání funkce polynomem je hodnota polynomu poloha a polynom je vždy pátého řádu a nelze to nijak změnit. `AccelerationScale` a `Offset` je samozřejmě pro zrychlení. Vzhledem ke komplikovaným výpočtům je doporučeno v tomto režimu vždy používat existující speciální grafický editor.

Poznámka 3: Dialog je jednotný pro `MC_AccelerationProfile`, `MC_VelocityProfile`, `MC_PositionProfile`. Některé režimy (parametr `alg` nedávají pro `AccelerationProfile` smysl a nefungují (aproximace B-spline nefunguje, režimy 1,2,5,6 používají lineární interpolaci)

Poznámka 4: Pokud na konci profilu je nenulová rychlost, osa se pohybuje dál touto rychlostí (to je v souladu se specifikací `PLCopen`).

Vstupy

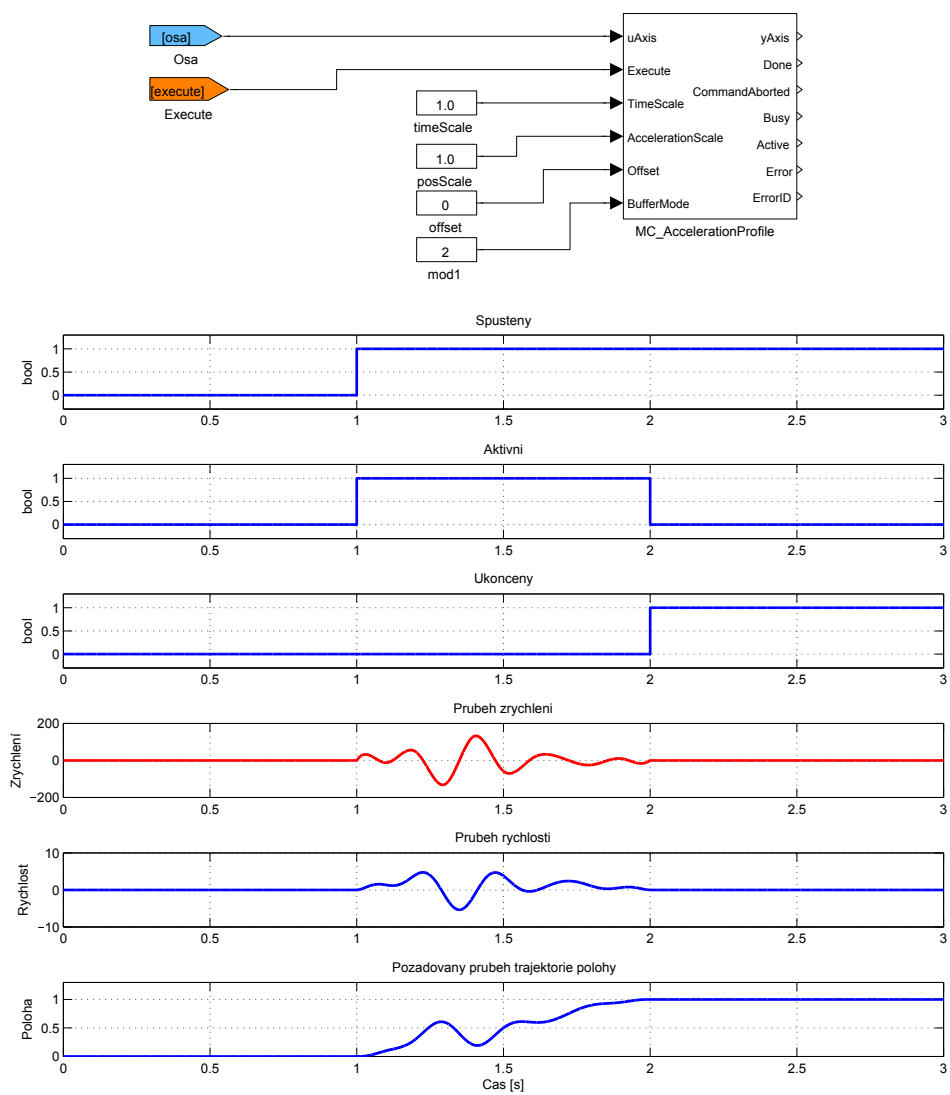
<code>uAxis</code>	Odkaz na osu (přípustné je jen spojení <code>RM_Axis.axisRef-uAxis</code> nebo <code>yAxis-uAxis</code>)	Reference
<code>Execute</code>	Náběžná hrana aktivuje blok	Bool
<code>alg</code>	Typ interpolace	⊙2 Long (I32)
	1 tabulka čas/hodnota (interpolace polynomem 3. řádu)	
	2 hodnoty ve stejném intervalu (interpolace polynomem 3. řádu)	
	3 interpolace polynomy 5. řádu	
	4 polynomy 5. řádu s ekvidistantními intervaly	
	5 tabulka čas/hodnota + počáteční a koncová derivace	
	6 hodnoty ve stejném intervalu + počáteční a koncová derivace	
	7 hodnoty ve stejném intervalu (aproximace B-splinem 3. řádu)	
	8 hodnoty ve stejném intervalu (aproximace B-splinem 5. řádu)	
	9 tabulka čas/hodnota (lineární interpolace)	
<code>nmax</code>	maximalni počet intervalů/bodů profilu	⊙3 Long (I32)
<code>TimeScale</code>	Konstanta násobení pro přepočet časové osy profilu	Double (F64)
<code>AccelerationScale</code>	Konstanta násobení pro přepočet hodnotové osy profilu	Double (F64)
<code>Offset</code>	Aditivní konstanta pro přepočet hodnotové osy profilu	Double (F64)
<code>BufferMode</code>	Režim převzetí osy	Long (I32)
<code>uTimes</code>	vektor s časovými hodnotami	Reference

uValues	vektor s hodnotami zrychlení nebo koeficienty polynomů	Reference
1	Aborting (nový blok se spustí okamžitě)	
2	Buffered (nový blok se spustí po dokončení předchozího)	
3	Blending low (nový blok se spustí po dokončení předchozího, původní pohyb skončí s nižší rychlostí z obou bloků)	
4	Blending high (nový blok se spustí po dokončení předchozího, původní pohyb skončí s vyšší rychlostí z obou bloků)	
5	Blending previous (nový blok se spustí po dokončení předchozího, původní pohyb skončí se svojí koncovou rychlostí)	
6	Blending next (nový blok se spustí po dokončení předchozího, původní pohyb skončí s počáteční rychlostí nového bloku)	

Výstupy

yAxis	Odkaz na osu (přípustné je jen spojení RM_Axis.axisRef-uAxis nebo yAxis-uAxis)	Reference
Done	Příznak dokončení algoritmu	Bool
CommandAborted	Příznak přerušení funkce bloku	Bool
Busy	Příznak, že algoritmus ještě neskončil	Bool
Active	Příznak, že blok řídí osu	Bool
Error	Příznak chyby	Bool
ErrorID	Výsledek poslední operace	Error
i	obecná chyba systému REXYGEN	

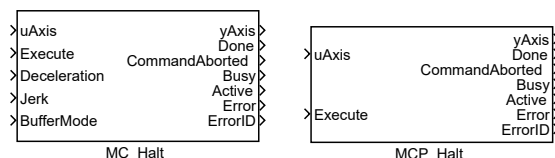
Příklad



MC_Halt, MCP_Halt – Zastavení pohybu (přerušitelné)

Symboly bloků

Licence: [MOTION CONTROL](#)



Popis funkce

Bloky MC_Halt a MCP_Halt mají naprosto shodnou funkci, jediným rozdílem je, že MCP_ varianta bloku má méně vstupů a potřebné konstanty se zadávají jako parametry bloku.

Blok **MC_Halt** zahajuje řízené zastavení pohybu. Osa se přesune do stavu DiscreteMotion, dokud není rychlost nulová. Společně s nastavením výstupu "Done" je stav změněn na "Standstill".

Poznámka 1: Blok **MC_Halt** se používá k zastavení osy za normálních provozních podmínek. V non-buffered režimu je možné zadat další pohybový příkaz při zpomalení osy, který zruší **MC_Halt** a bude ihned proveden.

Poznámka 2: Je-li tento příkaz aktivní, další příkaz může být aktivován (spuštěn). Např. vozidlo bez řidiče detekuje překážku a potřebuje zastavit. **MC_Halt** je aktivován. Před dosažením stavu "Standstill" je překážka odstraněna a pohyb může pokračovat nastavením dalšího pohybového příkazu, aby vozidlo nemuselo zastavit.

Vstupy

uAxis	Odkaz na osu (přípustné je jen spojení RM_Axis.axisRef-uAxis nebo yAxis-uAxis)	Reference
Execute	Náběžná hrana aktivuje blok	Bool
Deceleration	Maximální povolené zpomalení [unit/s ²]	Double (F64)
Jerk	Maximální povolená změna zrychlení [unit/s ³]	Double (F64)

Výstupy

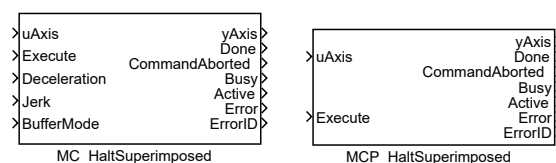
yAxis	Odkaz na osu (přípustné je jen spojení RM_Axis.axisRef-uAxis nebo yAxis-uAxis)	Reference
Done	Příznak dokončení algoritmu	Bool
CommandAborted	Příznak přerušení funkce bloku	Bool
Busy	Příznak, že algoritmus ještě neskončil	Bool
Active	Příznak, že blok řídí osu	Bool

Error	Příznak chyby	Bool
ErrorID	Výsledek poslední operace	Error
	i obecná chyba systému REXYGEN	

MC_HaltSuperimposed, MCP_HaltSuperimposed – Zastavení pohybu (přídavné a přerušitelné)

Symbody bloků

Licence: [MOTION CONTROL](#)



Popis funkce

Bloky MC_HaltSuperimposed a MCP_HaltSuperimposed mají naprosto shodnou funkci, jediným rozdílem je, že MCP_ varianta bloku má méně vstupů a potřebné konstanty se zadávají jako parametry bloku.

Blok MC_HaltSuperimposed zahajuje řízené zastavení přídavného pohybu, který generuje například blok [MC_MoveSuperimposed](#).

Vstupy

uAxis	Odkaz na osu (přípustné je jen spojení RM_Axis.axisRef-uAxis nebo yAxis-uAxis)	Reference
Execute	Náběžná hrana aktivuje blok	Bool
Deceleration	Maximální povolené zpomalení [unit/s ²]	Double (F64)
Jerk	Maximální povolená změna zrychlení [unit/s ³]	Double (F64)

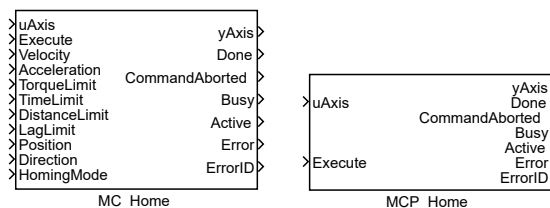
Výstupy

yAxis	Odkaz na osu (přípustné je jen spojení RM_Axis.axisRef-uAxis nebo yAxis-uAxis)	Reference
Done	Příznak dokončení algoritmu	Bool
CommandAborted	Příznak přerušení funkce bloku	Bool
Busy	Příznak, že algoritmus ještě neskončil	Bool
Active	Příznak, že blok řídí osu	Bool
Error	Příznak chyby	Bool
ErrorID	Výsledek poslední operace i obecná chyba systému REXYGEN	Error

MC_Home, MCP_Home – Nalezení výchozí polohy

Symboly bloků

Licence: MOTION CONTROL



Popis funkce

Bloky MC_Home a MCP_Home mají naprosto shodnou funkci, jediným rozdílem je, že MCP_ varianta bloku má méně vstupů a potřebné konstanty se zadávají jako parametry bloku.

Blok MC_Home provede algoritmus hledání výchozího bodu. Detaily závisí na parametrech bloku (zejména HomingMode). Podrobnější popis lze nalézt v PLCopen . Vstup Position je poloha, která je nastavena v okamžiku dosažení výchozí polohy. Po úspěšném ukončení algoritmu tohoto bloku je osa ve stavu „StandStill“.

Poznámka 1: Parametr či vstup BufferMode není podpořen - vždy je BufferMode = Aborting. To je nevýznamné omezení, protože tento blok se vykonává vždy jako první.

Poznámka 2: Tento blok vyžaduje připojené vstupy bloku RM_Axis. Podle zvolené metody může být potřeba ActualPos, ActualTorque, LimP, LimZ, LimN . Předpokládá se, že metoda je zvolena dopředu podle konstrukce stroje, proto není zvlášť vstup na nulový koncový spínač a referenční pulz - jeden z nich se připojí k LimZ a zvolí se HomingMode=3(nulový pulz).

Poznámka 3: HomingMode=4(uživatelská hodnota) pouze nastaví aktuální polohu. Proto také není implementován blok MC_SetPosition. HomingMode=5 (absolutní snímač polohy) pouze přepne osu do stavu StandStill.

Poznámka 4: Tento blok nepodporuje jerk (limit na derivaci zrychlení) a také není použit zvlášť limit na zrychlení a brždění. Pokud je potřeba nastavit výchozí polohu velmi přesně, doporučuje se spustit blok MC_Home dvakrát. Poprvé s velkou rychlostí pro najetí blízko výchozí polohy a podruhé s malou rychlostí pro přesné nastavení pozice.

Poznámka 5: Pro HomingMode=6 (najetí na mechanickou překážku) se požadovaná poloha pozná podle toho, že moment překročil nastavenou mez (tj. v tomto režimu to není chyba procedury) nebo že "position lag" překročil nastavenou mez (druhá podmínka se kontroluje jen pokud parametr MaxPositionLag v příslušném bloku RM_Axis je kladný).

Vstupy

<code>uAxis</code>	Odkaz na osu (přípustné je jen spojení <code>RM_Axis.axisRef-uAxis</code> nebo <code>yAxis-uAxis</code>)	Reference
<code>Execute</code>	Náběžná hrana aktivuje blok	Bool
<code>Velocity</code>	Maximální povolená rychlost [unit/s]	Double (F64)
<code>Acceleration</code>	Maximální povolené zrychlení [unit/s ²]	Double (F64)
<code>TorqueLimit</code>	Maximální povolený moment/síla	Double (F64)
<code>TimeLimit</code>	Maximální povolený čas pro celý algoritmus bloku [s]	Double (F64)
<code>DistanceLimit</code>	Maximální povolená vzdálenost pro celý algoritmus bloku [unit]	Double (F64)
<code>Position</code>	Požadovaná poloha [unit]	Double (F64)
<code>Direction</code>	Směr pohybu (jen pro cyckické osy nebo speciální případy)	Long (I32)
	1 kladný	
	2 nejkratší	
	3 záporný	
	4 aktuální	
<code>HomingMode</code>	Algoritmus hledání výchozí pozice	Long (I32)
	1 nulový spínač	
	2 koncové spínače	
	3 nulový pulz	
	4 uživatelská hodnota	
	5 absolutní snímač polohy	
	6 mechanický doraz	

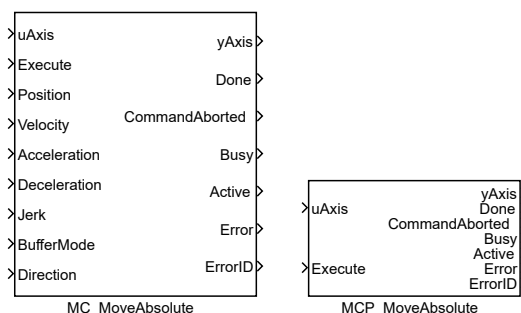
Výstupy

<code>yAxis</code>	Odkaz na osu (přípustné je jen spojení <code>RM_Axis.axisRef-uAxis</code> nebo <code>yAxis-uAxis</code>)	Reference
<code>Done</code>	Příznak dokončení algoritmu	Bool
<code>CommandAborted</code>	Příznak přerušení funkce bloku	Bool
<code>Busy</code>	Příznak, že algoritmus ještě neskončil	Bool
<code>Active</code>	Příznak, že blok řídí osu	Bool
<code>Error</code>	Příznak chyby	Bool
<code>ErrorID</code>	Výsledek poslední operace	Error
	i obecná chyba systému REXYGEN	

MC_MoveAbsolute, MCP_MoveAbsolute – Pohyb do pozice (absolutní souřadnice)

Symbyly bloků

Licence: [MOTION CONTROL](#)



Popis funkce

Bloky MC_MoveAbsolute a MCP_MoveAbsolute mají naprosto shodnou funkci, jediným rozdílem je, že MCP_ varianta bloku má méně vstupů a potřebné konstanty se zadávají jako parametry bloku.

Blok **MC_MoveAbsolute** přesune osu do zadané polohy za nejkratší možný čas (s respektováním zadaných omezení). Pokud není aktivován další blok, osa se v koncovém bodu zastaví. V opačném případě koncová rychlost závisí na parametru **BufferedMode** následujícího bloku (viz popis tohoto parametru). Pro potřeby blending módu je počáteční a koncová rychlost tohoto bloku rovna jeho maximální dovolené rychlosti (tj. parametr **Velocity**). Pokud je směr následujícího bloku opačný, provede se přepnutí při nulové rychlosti (tj. stejně jako pro mod buffered).

Pokud je použit blending mod a následující blok je spuštěn příliš pozdě, může se stát, že není možné dosáhnout požadované rychlosti. Je několik způsobů, jak toto řešit:

1. První blok skončí s chybou a osa přejde do chybového stavu.
2. První blok skončí s chybou a řízení osy okamžitě přebírá následující blok.
3. První blok s respektováním omezení na maximální zrychlení a jerk vygeneruje trajektorii, která se co nejvíce blíží požadovanému koncovému bodu. K přepnutí řízení na následující blok dojde ve správné pozici, ale při jiné rychlosti, než je požadováno.
4. První blok „zabrzdí“, „kousek se vrátí“ a dokončí pohyb tak, že k přepnutí dojde v cílové poloze a s požadovanou rychlostí.
5. První blok blending mod ignoruje a dokončí pohyb, tj. chová se stejně, jako pro mod buffered.

Každá z uvedených variant má určité výhody a nevýhody. V současnosti je použita varianta 3 pro případ bez omezení jerku a varianta 4 pro případ s omezením jerku. Nicméně

z hlediska návrhu aplikace je potřeba uvedenou situaci považovat za nedefinovaný stav a vyhnout se jí.

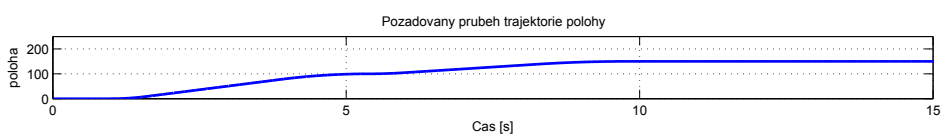
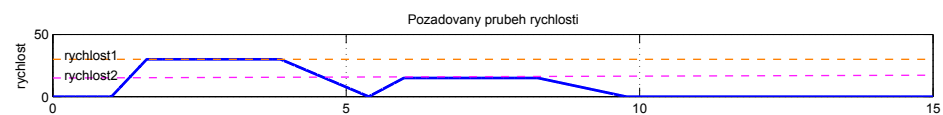
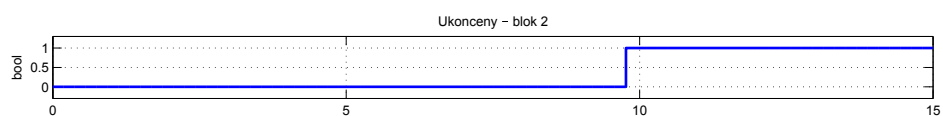
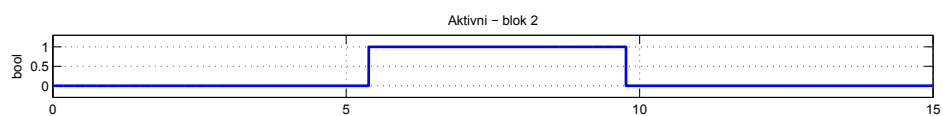
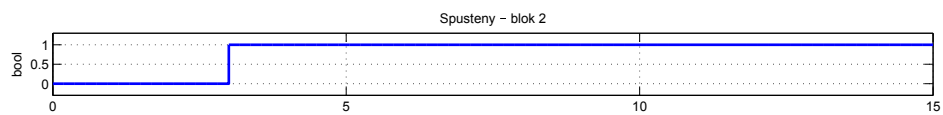
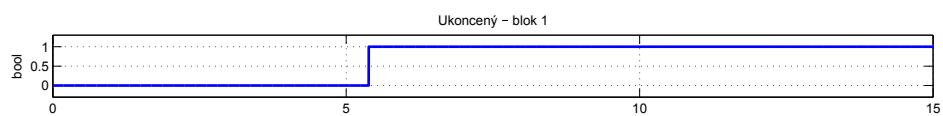
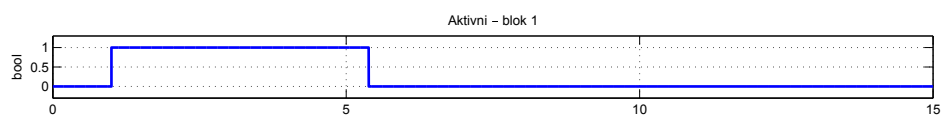
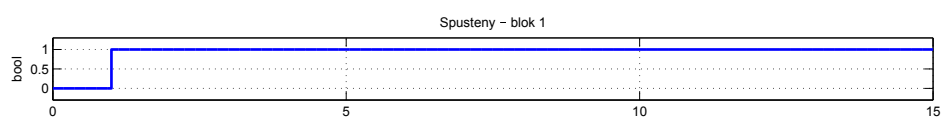
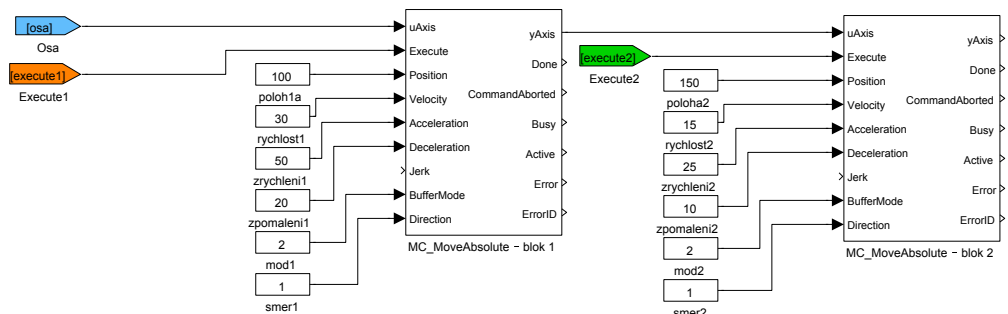
Vstupy

uAxis	Odkaz na osu (přípustné je jen spojení <code>RM_Axis.axisRef-uAxis</code> nebo <code>yAxis-uAxis</code>)	Reference
Execute	Náběžná hrana aktivuje blok	Bool
Position	Požadovaná poloha [unit]	Double (F64)
Velocity	Maximální povolená rychlost [unit/s]	Double (F64)
Acceleration	Maximální povolené zrychlení [unit/s ²]	Double (F64)
Deceleration	Maximální povolené zpomalení [unit/s ²]	Double (F64)
Jerk	Maximální povolená změna zrychlení [unit/s ³]	Double (F64)
BufferMode	Režim převzetí osy	Long (I32)
	1 Aborting (nový blok se spustí okamžitě)	
	2 Buffered (nový blok se spustí po dokončení předchozího)	
	3 Blending low (nový blok se spustí po dokončení předchozího, původní pohyb skončí s nižší rychlostí z obou bloků)	
	4 Blending high (nový blok se spustí po dokončení předchozího, původní pohyb skončí s vyšší rychlostí z obou bloků)	
	5 Blending previous (nový blok se spustí po dokončení předchozího, původní pohyb skončí se svojí koncovou rychlostí)	
	6 Blending next (nový blok se spustí po dokončení předchozího, původní pohyb skončí s počáteční rychlostí nového bloku)	
Direction	Směr pohybu (jen pro cyklické osy nebo speciální případy)	Long (I32)
	1 kladný	
	2 nejkratší	
	3 záporný	
	4 aktuální	

Výstupy

yAxis	Odkaz na osu (přípustné je jen spojení <code>RM_Axis.axisRef-uAxis</code> nebo <code>yAxis-uAxis</code>)	Reference
Done	Příznak dokončení algoritmu	Bool
CommandAborted	Příznak přerušení funkce bloku	Bool
Busy	Příznak, že algoritmus ještě neskončil	Bool
Active	Příznak, že blok řídí osu	Bool
Error	Příznak chyby	Bool
ErrorID	Výsledek poslední operace	Error
	i obecná chyba systému REXYGEN	

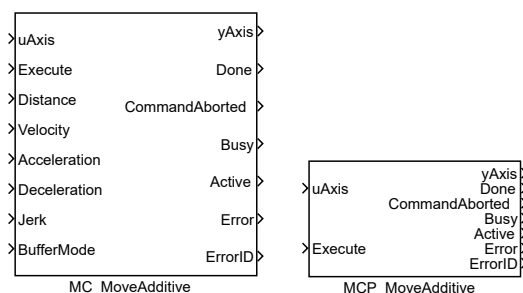
Příklad



MC_MoveAdditive, MCP_MoveAdditive – Pohyb do pozice (relativně ke konci předchozího pohybu)

Symbyly bloků

Licence: [MOTION CONTROL](#)



Popis funkce

Bloky MC_MoveAbsolute a MCP_MoveAbsolute mají naprosto shodnou funkci, jediným rozdílem je, že MCP_ varianta bloku má méně vstupů a potřebné konstanty se zadávají jako parametry bloku.

Blok **MC_MoveAdditive** přesune osu do zadané polohy za nejkratší možný čas (s respektováním zadaných omezení). Koncová poloha se určí tak, že se k aktuální poloze v okamžiku převzetí řízení osy přičte hodnota parametru **Distance**. Pokud není aktivován další blok, osa se v koncovém bodu zastaví. V opačném případě koncová rychlost závisí na parametru **BufferedMode** následujícího bloku (viz popis tohoto parametru). Pro potřeby "blending" módu je počáteční a koncová rychlost tohoto bloku rovna jeho maximální dovolené rychlosti (tj. parametr **Velocity**). Pokud je směr následujícího bloku opačný, provede se přepnutí při nulové rychlosti (tj. stejně jako pro mód buffered).

Pokud je použit blending mod a následující blok je spuštěn příliš pozdě, může se stát, že není možné dosáhnout požadované rychlosti. Je několik způsobů, jak toto řešit:

1. První blok skončí s chybou a osa přejde do chybového stavu.
2. První blok skončí s chybou a řízení osy okamžitě přebírá následující blok.
3. První blok s respektováním omezení na maximální zrychlení a jerk vygeneruje trajektorii, která se co nejvíce blíží požadovanému koncovému bodu. K přepnutí řízení na následující blok dojde ve správné pozici, ale při jiné rychlosti, než je požadováno.
4. První blok „zabrzdí“, „kousek se vrátí“ a dokončí pohyb tak, že k přepnutí dojde v cílové poloze a s požadovanou rychlostí.
5. První blok blending mod ignoruje a dokončí pohyb, tj. chová se stejně, jako pro mod buffered.

Každá z uvedených variant má určité výhody a nevýhody. V současnosti je použita va-

rianta 3 pro případ bez omezení jerku a varianta 4 pro případ s omezením jerku. Nicméně z hlediska návrhu aplikace je potřeba uvedenou situaci považovat za nedefinovaný stav a vyhnout se jí.

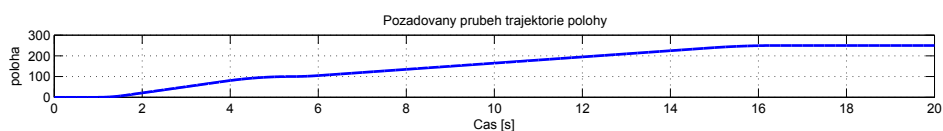
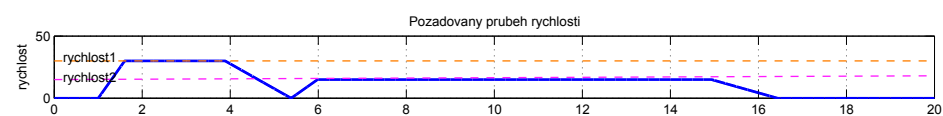
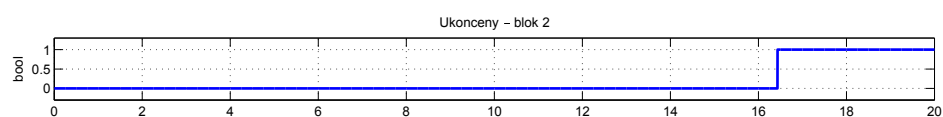
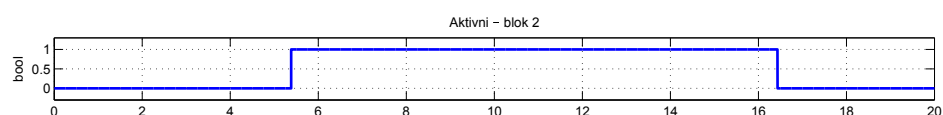
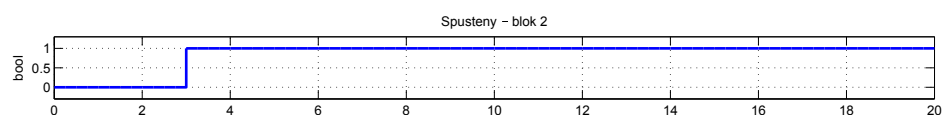
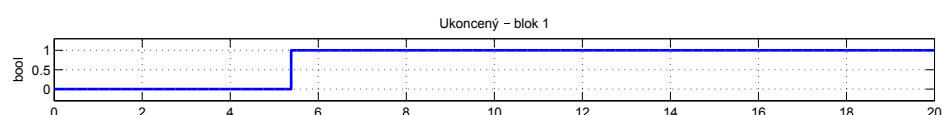
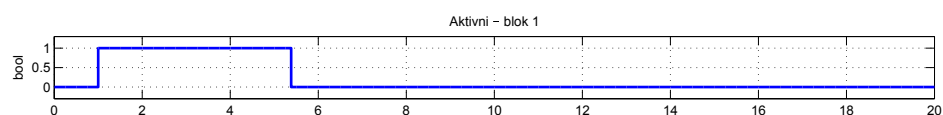
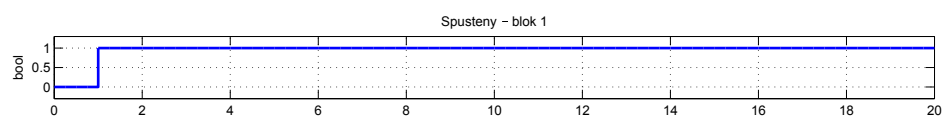
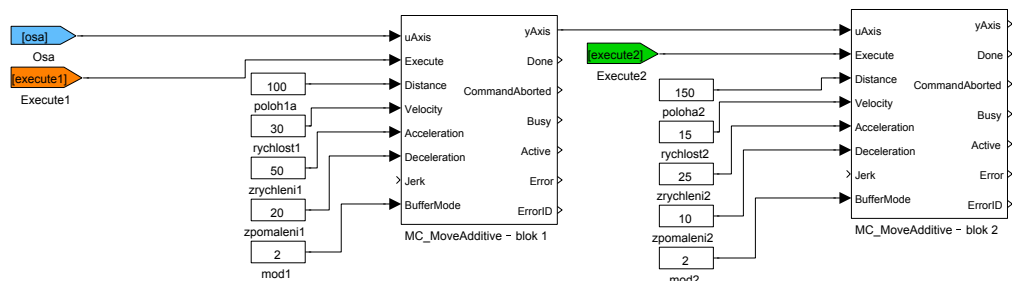
Vstupy

uAxis	Odkaz na osu (přípustné je jen spojení <code>RM_Axis.axisRef-uAxis</code> nebo <code>yAxis-uAxis</code>)	Reference
Execute	Náběžná hrana aktivuje blok	Bool
Distance	Požadovaná vzdálenost (od okamžiku začátku řízení osy blokem) [unit]	Double (F64)
Velocity	Maximální povolená rychlost [unit/s]	Double (F64)
Acceleration	Maximální povolené zrychlení [unit/s ²]	Double (F64)
Deceleration	Maximální povolené zpomalení [unit/s ²]	Double (F64)
Jerk	Maximální povolená změna zrychlení [unit/s ³]	Double (F64)
BufferMode	Režim převzetí osy	Long (I32)
	1 Aborting (nový blok se spustí okamžitě)	
	2 Buffered (nový blok se spustí po dokončení předchozího)	
	3 Blending low (nový blok se spustí po dokončení předchozího, původní pohyb skončí s nižší rychlostí z obou bloků)	
	4 Blending high (nový blok se spustí po dokončení předchozího, původní pohyb skončí s vyšší rychlostí z obou bloků)	
	5 Blending previous (nový blok se spustí po dokončení předchozího, původní pohyb skončí se svojí koncovou rychlostí)	
	6 Blending next (nový blok se spustí po dokončení předchozího, původní pohyb skončí s počáteční rychlostí nového bloku)	

Výstupy

yAxis	Odkaz na osu (přípustné je jen spojení <code>RM_Axis.axisRef-uAxis</code> nebo <code>yAxis-uAxis</code>)	Reference
Done	Příznak dokončení algoritmu	Bool
CommandAborted	Příznak přerušení funkce bloku	Bool
Busy	Příznak, že algoritmus ještě neskončil	Bool
Active	Příznak, že blok řídí osu	Bool
Error	Příznak chyby	Bool
ErrorID	Výsledek poslední operace	Error
	i obecná chyba systému REXYGEN	

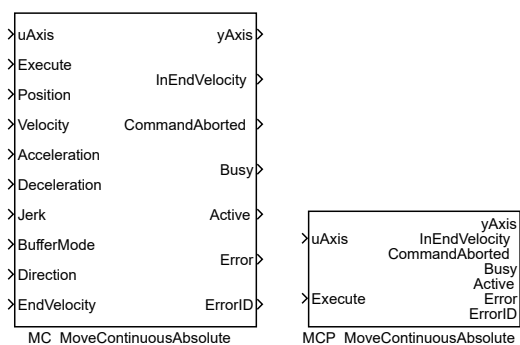
Příklad



MC_MoveContinuousAbsolute, MCP_MoveContinuousAbsolute – Pohyb do pozice (absolutní souřadnice)

Symboly bloků

Licence: [MOTION CONTROL](#)



Popis funkce

Bloky MC_MoveAbsolute a MCP_MoveAbsolute mají naprosto shodnou funkci, jediným rozdílem je, že MCP_ varianta bloku má méně vstupů a potřebné konstanty se zadávají jako parametry bloku.

Blok **MC_MoveContinuousAbsolute** přesune osu do zadané polohy za nejkratší možný čas (s respektováním zadaných omezení). Pokud není aktivován další blok, koncová rychlost zůstává konstantní na hodnotě parametru **EndVelocity**, osa se tedy nezastaví. V opačném případě koncová rychlost závisí na parametru **BufferedMode** následujícího bloku (viz popis tohoto parametru). Pro potřeby blending módu je počáteční a koncová rychlost tohoto bloku rovna jeho maximální dovolené rychlosti (tj. parametr **Velocity**). Pokud je směr následujícího bloku opačný, provede se přepnutí při nulové rychlosti.

Pokud je použit blending mod a následující blok je spuštěn příliš pozdě, může se stát, že není možné dosáhnout požadované rychlosti. Je několik způsobů, jak toto řešit:

1. První blok skončí s chybou a osa přejde do chybového stavu.
2. První blok skončí s chybou a řízení osy okamžitě přebírá následující blok.
3. První blok s respektováním omezení na maximální zrychlení a jerk vygeneruje trajektorii, která se co nejvíce blíží požadovanému koncovému bodu. K přepnutí řízení na následující blok dojde ve správné pozici, ale při jiné rychlosti, než je požadováno.
4. První blok „zabrzdí“, „kousek se vrátí“ a dokončí pohyb tak, že k přepnutí dojde v cílové poloze a s požadovanou rychlostí.
5. První blok blending mod ignoruje a dokončí pohyb, tj. chová se stejně, jako pro mod buffered.

Každá z uvedených variant má určité výhody a nevýhody. V současnosti je použita varianta 3 pro případ bez omezení jerku a varianta 4 pro případ s omezením jerku. Nicméně z hlediska návrhu aplikace je potřeba uvedenou situaci považovat za nedefinovaný stav a vyhnout se jí.

Poznámka 1: Jestliže je nastaven parametr **EndVelocity** na nulovou hodnotu, pak se chová blok stejným způsobem jako **MC_MoveAbsolute**.

Poznámka 2: Pokud dojde ke spuštění dalšího bloku dříve, než je dosaženo požadované polohy, blok se opět chová stejným způsobem jako **MC_MoveAbsolute**.

Vstupy

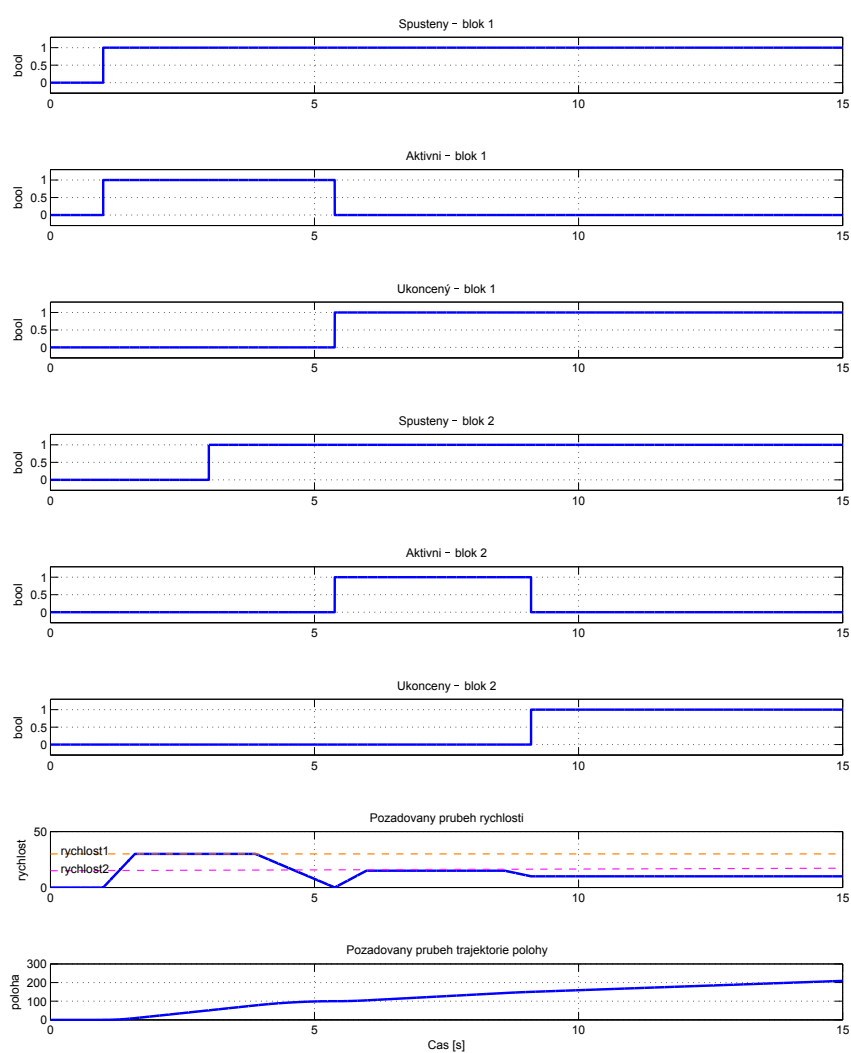
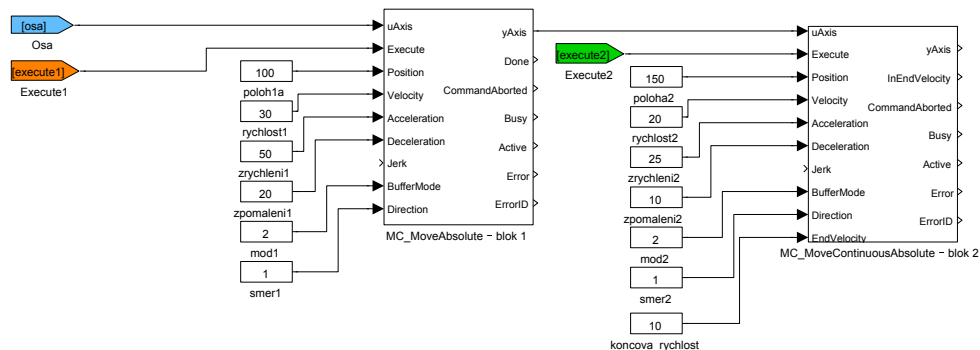
uAxis	Odkaz na osu (přípustné je jen spojení RM_Axis.axisRef-uAxis nebo yAxis-uAxis)	Reference
Execute	Náběžná hrana aktivuje blok	Bool
Position	Požadovaná poloha [unit]	Double (F64)
Velocity	Maximální povolená rychlost [unit/s]	Double (F64)
Acceleration	Maximální povolené zrychlení [unit/s ²]	Double (F64)
Deceleration	Maximální povolené zpomalení [unit/s ²]	Double (F64)
Jerk	Maximální povolená změna zrychlení [unit/s ³]	Double (F64)
BufferMode	Režim převzetí osy	Long (I32)
	1 Aborting (nový blok se spustí okamžitě)	
	2 Buffered (nový blok se spustí po dokončení předchozího)	
	3 Blending low (nový blok se spustí po dokončení předchozího, původní pohyb skončí s nižší rychlostí z obou bloků)	
	4 Blending high (nový blok se spustí po dokončení předchozího, původní pohyb skončí s vyšší rychlostí z obou bloků)	
	5 Blending previous (nový blok se spustí po dokončení předchozího, původní pohyb skončí se svojí koncovou rychlostí)	
	6 Blending next (nový blok se spustí po dokončení předchozího, původní pohyb skončí s počáteční rychlostí nového bloku)	
Direction	Směr pohybu (jen pro cyklické osy nebo speciální případy)	Long (I32)
	1 kladný	
	2 nejkratší	
	3 záporný	
	4 aktuální	
EndVelocity	Koncová rychlost	Double (F64)

Výstupy

yAxis	Odkaz na osu (přípustné je jen spojení RM_Axis.axisRef-uAxis nebo yAxis-uAxis)	Reference
InEndVelocity	Příznak dokončení algoritmu	Bool

CommandAborted	Příznak přerušení funkce bloku	Bool
Busy	Příznak, že algoritmus ještě neskončil	Bool
Active	Příznak, že blok řídí osu	Bool
Error	Příznak chyby	Bool
ErrorID	Výsledek poslední operace	Error
	i obecná chyba systému REXYGEN	

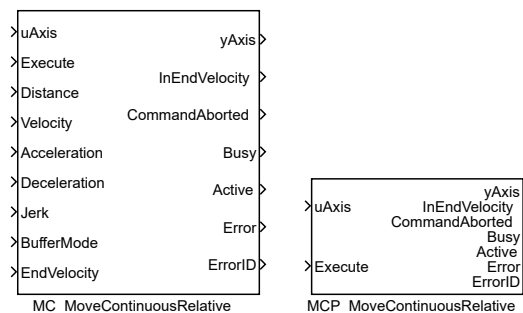
Příklad



MC_MoveContinuousRelative, MCP_MoveContinuousRelative – Pohyb do pozice (relativně ke konci předchozího pohybu)

Symbole bloků

Licence: [MOTION CONTROL](#)



Popis funkce

Bloky MC_MoveContinuousRelative a MCP_MoveContinuousRelative mají naprosto shodnou funkci, jediným rozdílem je, že MCP_ varianta bloku má méně vstupů a potřebné konstanty se zadávají jako parametry bloku.

Blok **MC_MoveContinuousRelative** přesune osu do zadané polohy za nejkratší možný čas (s respektováním zadaných omezení). Koncová poloha se určí tak, že se k aktuální poloze v okamžiku spuštění (tj. náběžné hrany na vstupu **Execute**) přičte hodnota parametru **Distance**. Pokud není aktivován další blok, koncová rychlost zůstává konstantní na hodnotě parametru **EndVelocity**, osa se tedy nezastaví. V opačném případě koncová rychlost závisí na parametru **BufferedMode** následujícího bloku (viz popis tohoto parametru). Pro potřeby "blending" módu je počáteční a koncová rychlost tohoto bloku rovna jeho maximální dovolené rychlosti (tj. parametr **Velocity**). Pokud je směr následujícího bloku opačný, provede se přepnutí při nulové rychlosti (tj. stejně jako pro mod buffered).

Pokud je použit blending mod a následující blok je spuštěn příliš pozdě, může se stát, že není možné dosáhnout požadované rychlosti. Je několik způsobů, jak toto řešit:

1. První blok skončí s chybou a osa přejde do chybového stavu.
2. První blok skončí s chybou a řízení osy okamžitě přebírá následující blok.
3. První blok s respektováním omezení na maximální zrychlení a jerk vygeneruje trajektorii, která se co nejvíce blíží požadovanému koncovému bodu. K přepnutí řízení na následující blok dojde ve správné pozici, ale při jiné rychlosti, než je požadováno.
4. První blok „zabrzdí“, „kousek se vrátí“ a dokončí pohyb tak, že k přepnutí dojde v cílové poloze a s požadovanou rychlostí.
5. První blok blending mod ignoruje a dokončí pohyb, tj. chová se stejně, jako pro mod buffered.

Každá z uvedených variant má určité výhody a nevýhody. V současnosti je použita varianta 3 pro případ bez omezení jerku a varianta 4 pro případ s omezením jerku. Nicméně z hlediska návrhu aplikace je potřeba uvedenou situaci považovat za nedefinovaný stav a vyhnout se jí.

Poznámka 1: Jestliže je nastaven parametr **EndVelocity** na nulovou hodnotu, pak se chová blok stejným způsobem jako **MC_MoveRelative**.

Poznámka 2: Pokud dojde ke spuštění dalšího bloku dříve, než je dosaženo požadované polohy, blok se opět chová stejným způsobem jako **MC_MoveRelative**.

Vstupy

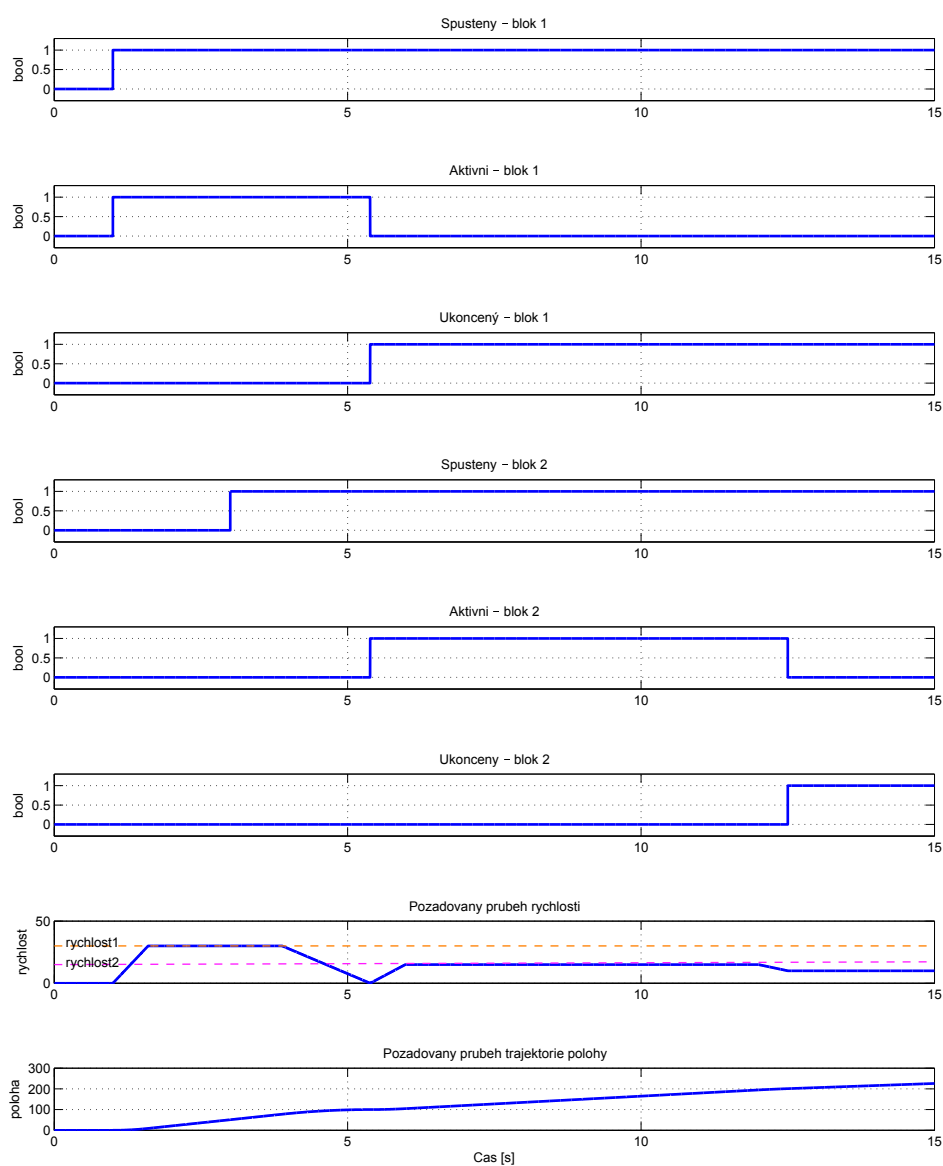
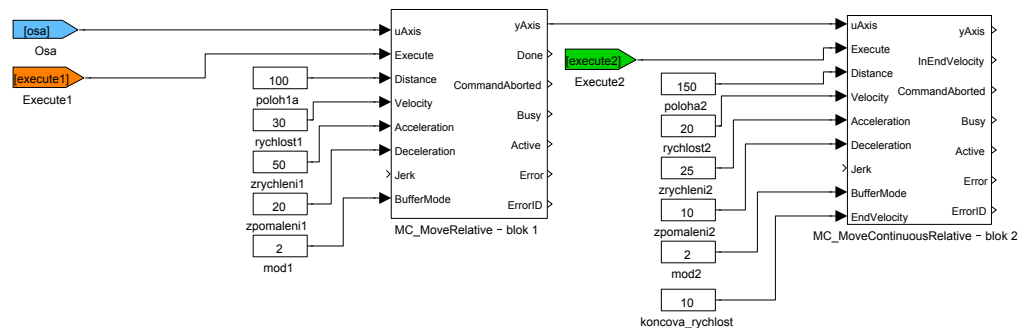
uAxis	Odkaz na osu (přípustné je jen spojení RM_Axis.axisRef-uAxis nebo yAxis-uAxis)	Reference
Execute	Náběžná hrana aktivuje blok	Bool
Distance	Požadovaná vzdálenost (od okamžiku startu bloku) [unit]	Double (F64)
Velocity	Maximální povolená rychlost [unit/s]	Double (F64)
Acceleration	Maximální povolené zrychlení [unit/s ²]	Double (F64)
Deceleration	Maximální povolené zpomalení [unit/s ²]	Double (F64)
Jerk	Maximální povolená změna zrychlení [unit/s ³]	Double (F64)
BufferMode	Režim převzetí osy	Long (I32)
	1 Aborting (nový blok se spustí okamžitě)	
	2 Buffered (nový blok se spustí po dokončení předchozího)	
	3 Blending low (nový blok se spustí po dokončení předchozího, původní pohyb skončí s nižší rychlostí z obou bloků)	
	4 Blending high (nový blok se spustí po dokončení předchozího, původní pohyb skončí s vyšší rychlostí z obou bloků)	
	5 Blending previous (nový blok se spustí po dokončení předchozího, původní pohyb skončí se svojí koncovou rychlostí)	
	6 Blending next (nový blok se spustí po dokončení předchozího, původní pohyb skončí s počáteční rychlostí nového bloku)	
EndVelocity	Koncová rychlost	Long (I32)

Výstupy

yAxis	Odkaz na osu (přípustné je jen spojení RM_Axis.axisRef-uAxis nebo yAxis-uAxis)	Reference
InEndVelocity	Příznak dokončení algoritmu	Bool
CommandAborted	Příznak přerušení funkce bloku	Bool
Busy	Příznak, že algoritmus ještě neskončil	Bool
Active	Příznak, že blok řídí osu	Bool
Error	Příznak chyby	Bool

ErrorID	Výsledek poslední operace	Error
i	 obecná chyba systému REXYGEN	

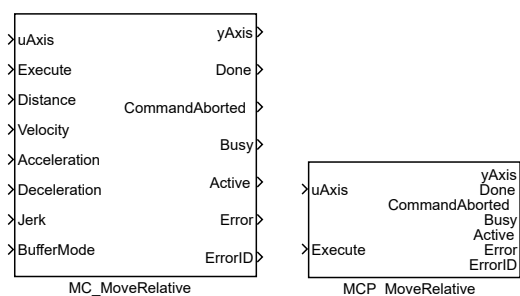
Příklad



MC_MoveRelative, MCP_MoveRelative – Pohyb do pozice (relativně k okamžiku spuštění)

Symbylo bloků

Licence: [MOTION CONTROL](#)



Popis funkce

Bloky MC_MoveRelative a MCP_MoveRelative mají naprosto shodnou funkci, jediným rozdílem je, že MCP_ varianta bloku má méně vstupů a potřebné konstanty se zadávají jako parametry bloku.

Blok MC_MoveRelative přesune osu do zadané polohy za nejkratší možný čas (s respektováním zadaných omezení). Koncová poloha se určí tak, že se k aktuální poloze v okamžiku spuštění (tj. náběžné hrany na vstupu **Execute**) přičte hodnota parametru **Distance**. Pokud není aktivován další blok, osa se v koncovém bodu zastaví. V opačném případě koncová rychlost závisí na parametru **BufferedMode** následujícího bloku (viz popis tohoto parametru). Pro potřeby "blending" módu je počáteční a koncová rychlost tohoto bloku rovna jeho maximální dovolené rychlosti (tj. parametr **Velocity**). Pokud je směr následujícího bloku opačný, provede se přepnutí při nulové rychlosti (tj. stejně jako pro mod buffered).

Pokud je použit blending mod a následující blok je spuštěn příliš pozdě, může se stát, že není možné dosáhnout požadované rychlosti. Je několik způsobů, jak toto řešit:

1. První blok skončí s chybou a osa přejde do chybového stavu.
2. První blok skončí s chybou a řízení osy okamžitě přebírá následující blok.
3. První blok s respektováním omezení na maximální zrychlení a jerk vygeneruje trajektorii, která se co nejvíce blíží požadovanému koncovému bodu. K přepnutí řízení na následující blok dojde ve správné pozici, ale při jiné rychlosti, než je požadováno.
4. První blok „zabrzdí“, „kousek se vrátí“ a dokončí pohyb tak, že k přepnutí dojde v cílové poloze a s požadovanou rychlostí.
5. První blok blending mod ignoruje a dokončí pohyb, tj. chová se stejně, jako pro mod buffered.

Každá z uvedených variant má určité výhody a nevýhody. V současnosti je použita varianta 3 pro případ bez omezení jerku a varianta 4 pro případ s omezením jerku. Nicméně z hlediska návrhu aplikace je potřeba uvedenou situaci považovat za nedefinovaný stav a vyhnout se jí.

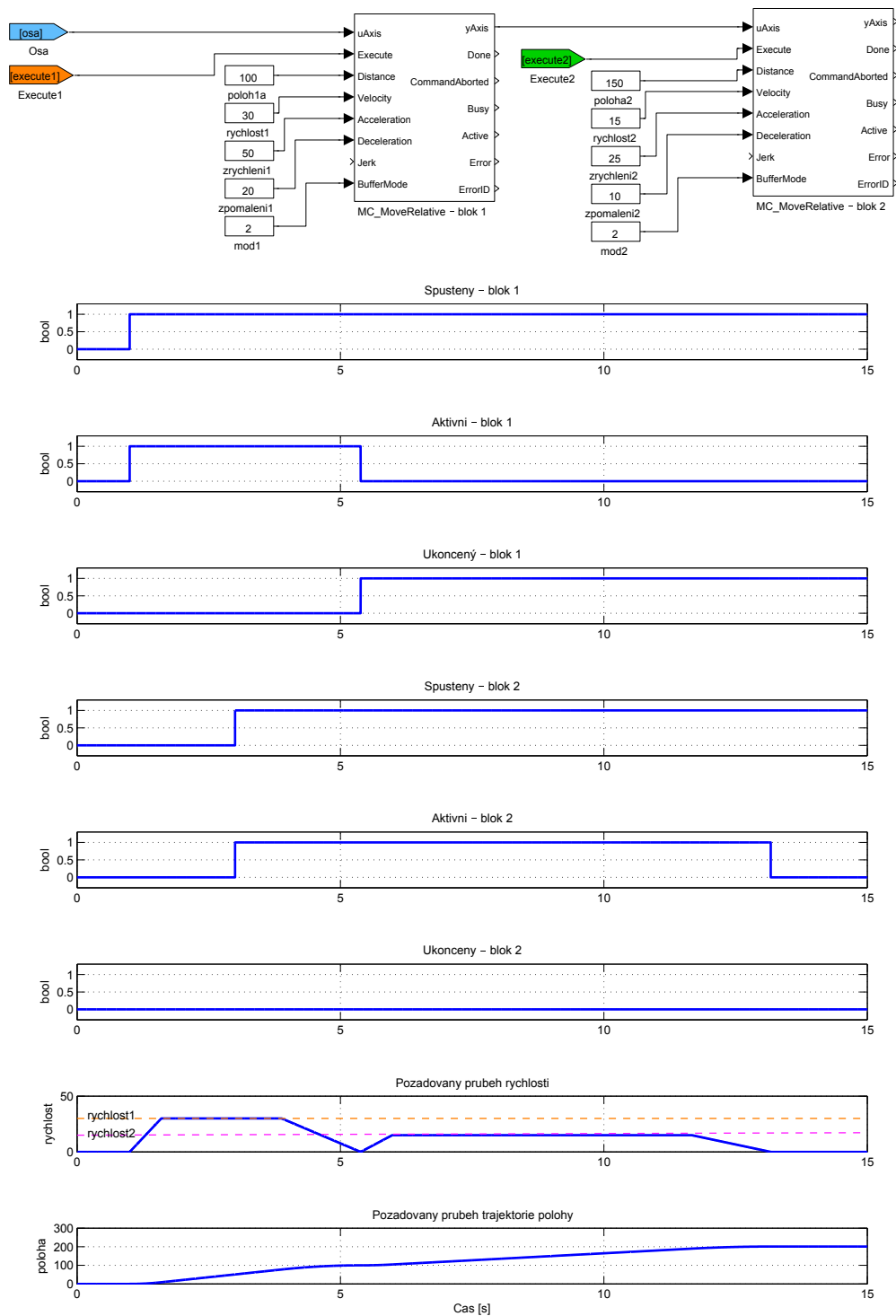
Vstupy

uAxis	Odkaz na osu (přípustné je jen spojení <code>RM_Axis.axisRef-uAxis</code> nebo <code>yAxis-uAxis</code>)	Reference
Execute	Náběžná hrana aktivuje blok	Bool
Distance	Požadovaná vzdálenost (od okamžiku startu bloku) [unit]	Double (F64)
Velocity	Maximální povolená rychlost [unit/s]	Double (F64)
Acceleration	Maximální povolené zrychlení [unit/s ²]	Double (F64)
Deceleration	Maximální povolené zpomalení [unit/s ²]	Double (F64)
Jerk	Maximální povolená změna zrychlení [unit/s ³]	Double (F64)
BufferMode	Režim převzetí osy	Long (I32)
	1 Aborting (nový blok se spustí okamžitě)	
	2 Buffered (nový blok se spustí po dokončení předchozího)	
	3 Blending low (nový blok se spustí po dokončení předchozího, původní pohyb skončí s nižší rychlostí z obou bloků)	
	4 Blending high (nový blok se spustí po dokončení předchozího, původní pohyb skončí s vyšší rychlostí z obou bloků)	
	5 Blending previous (nový blok se spustí po dokončení předchozího, původní pohyb skončí se svojí koncovou rychlostí)	
	6 Blending next (nový blok se spustí po dokončení předchozího, původní pohyb skončí s počáteční rychlostí nového bloku)	

Výstupy

yAxis	Odkaz na osu (přípustné je jen spojení <code>RM_Axis.axisRef-uAxis</code> nebo <code>yAxis-uAxis</code>)	Reference
Done	Příznak dokončení algoritmu	Bool
CommandAborted	Příznak přerušení funkce bloku	Bool
Busy	Příznak, že algoritmus ještě neskončil	Bool
Active	Příznak, že blok řídí osu	Bool
Error	Příznak chyby	Bool
ErrorID	Výsledek poslední operace	Error
	i obecná chyba systému REXYGEN	

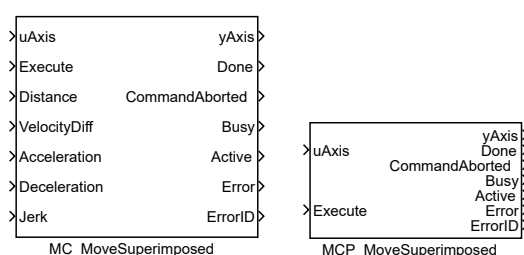
Příklad



MC_MoveSuperimposed, MCP_MoveSuperimposed – Pohyb do pozice (přídavný pohyb)

Symbyly bloků

Licence: [MOTION CONTROL](#)



Popis funkce

Bloky MC_MoveSuperimposed a MCP_MoveSuperimposed mají naprosto shodnou funkci, jediným rozdílem je, že MCP_ varianta bloku má méně vstupů a potřebné konstanty se zadávají jako parametry bloku.

Blok MC_MoveSuperimposed přesune osu do zadané polohy za nejkratší možný čas (s respektováním zadaných omezení). Koncová poloha je hodnota parametru **Distance**, přičemž počáteční poloha se považuje za nulovou. Pokud již nějaký blok běží, původní blok běží dále a hodnoty (poloha, rychlost, zrychlení) superimposed bloku se přičítají k hodnotám generovaným původním blokem. Pokud žádný blok neběží, tento blok se chová stejně jako [MC_MoveRelative](#).

Poznámka: Tento blok nemá parametr **BufferMode**, protože v superimposed režimu je to irrelevantní. Pokud v okamžiku spuštění (náběžná hrana na vstupu **Execute**) je již nějaký blok v režimu superimposed aktivní, počká se na doběhnutí a pak se spustí nový blok (tj. analogicky režimu buffered).

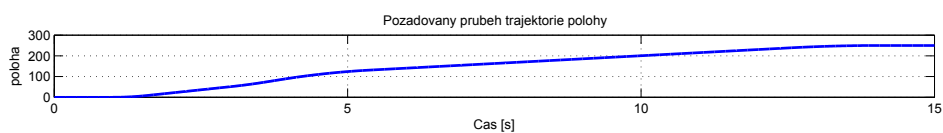
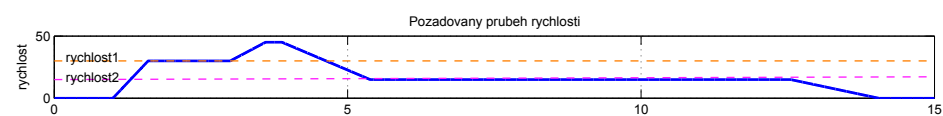
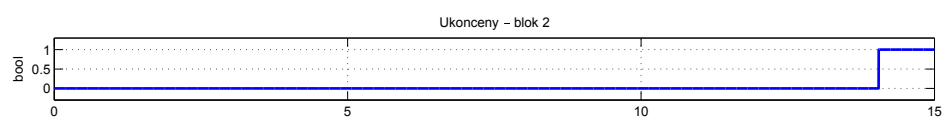
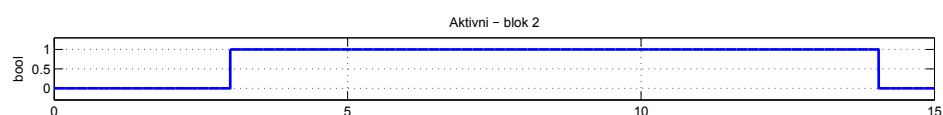
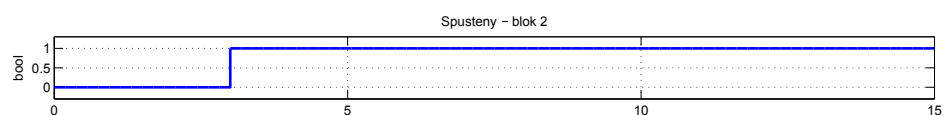
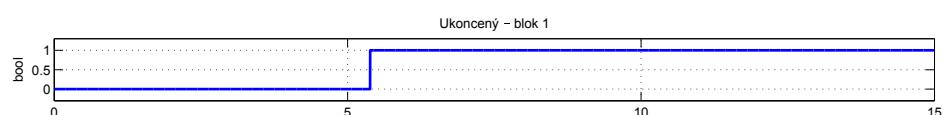
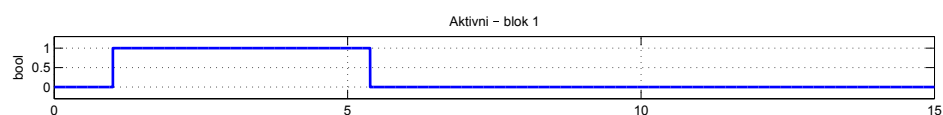
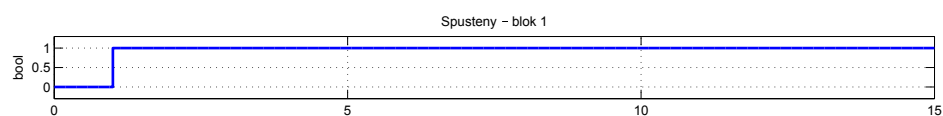
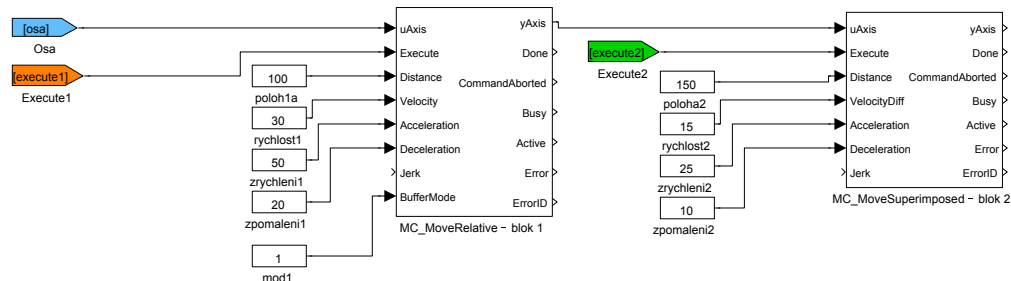
Vstupy

uAxis	Odkaz na osu (přípustné je jen spojení <code>RM_Axis.axisRef-uAxis</code> nebo <code>yAxis-uAxis</code>)	Reference
Execute	Náběžná hrana aktivuje blok	Bool
Distance	Požadovaná vzdálenost (od okamžiku startu bloku) [unit]	Double (F64)
VelocityDiff	Maximální povolená rychlost [unit/s]	Double (F64)
Acceleration	Maximální povolené zrychlení [unit/s ²]	Double (F64)
Deceleration	Maximální povolené zpomalení [unit/s ²]	Double (F64)
Jerk	Maximální povolená změna zrychlení [unit/s ³]	Double (F64)

Výstupy

yAxis	Odkaz na osu (přípustné je jen spojení RM_Axis.axisRef–uAxis nebo yAxis–uAxis)	Reference
Done	Příznak dokončení algoritmu	Bool
CommandAborted	Příznak přerušení funkce bloku	Bool
Busy	Příznak, že algoritmus ještě neskončil	Bool
Active	Příznak, že blok řídí osu	Bool
Error	Příznak chyby	Bool
ErrorID	Výsledek poslední operace i obecná chyba systému REXYGEN	Error

Příklad

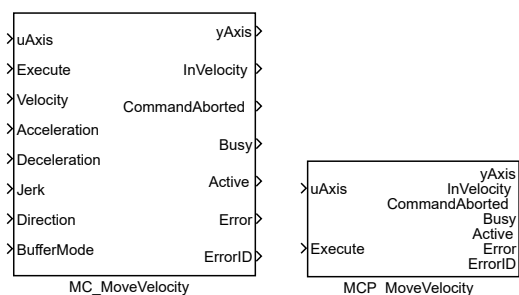


Cas [s]

MC_MoveVelocity, MCP_MoveVelocity – Pohyb konstantní rychlostí

Symboly bloků

Licence: [MOTION CONTROL](#)



Popis funkce

Bloky MC_MoveVelocity a MCP_MoveVelocity mají naprosto shodnou funkci, jediným rozdílem je, že MCP_ varianta bloku má méně vstupů a potřebné konstanty se zadávají jako parametry bloku.

Blok **MC_MoveVelocity** změní rychlost osy na požadovanou hodnotu za nejkratší možný čas s respektováním omezení na zrychlení a popřípadě jerk. Rychlost pak zůstává konstantní, dokud není aktivován jiný blok.

Vstupy

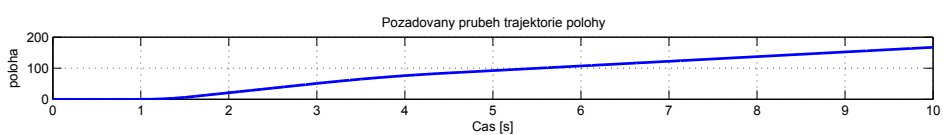
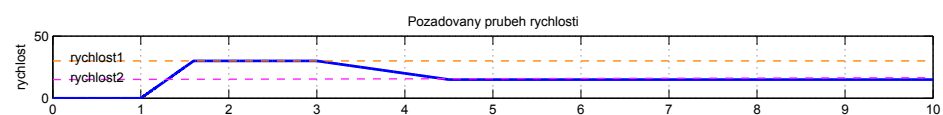
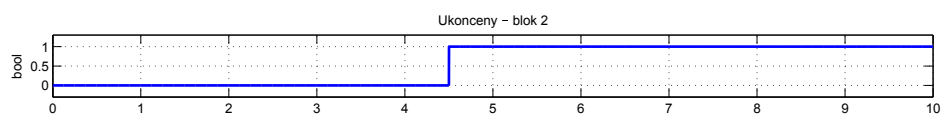
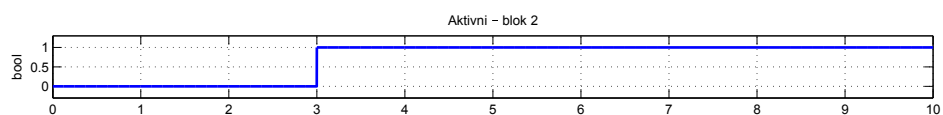
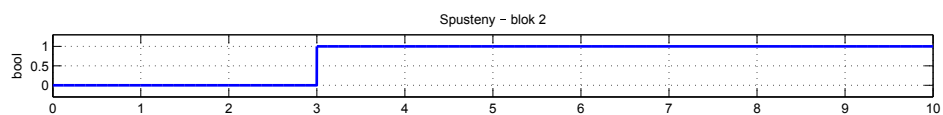
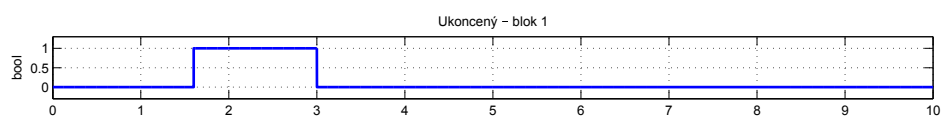
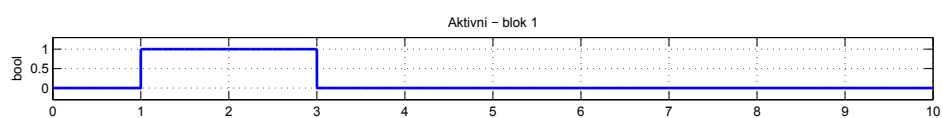
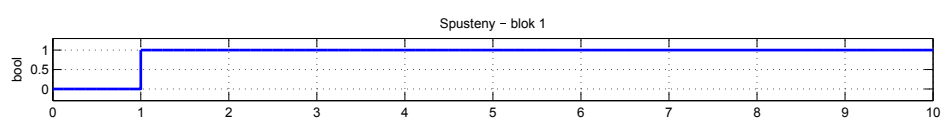
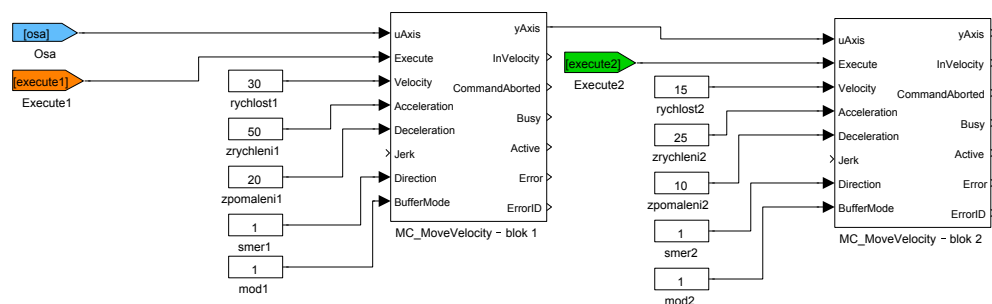
uAxis	Odkaz na osu (přípustné je jen spojení <code>RM_Axis.axisRef-uAxis</code> nebo <code>yAxis-uAxis</code>)	Reference
Execute	Náběžná hrana aktivuje blok	Bool
Velocity	Maximální povolená rychlost [unit/s]	Double (F64)
Acceleration	Maximální povolené zrychlení [unit/s ²]	Double (F64)
Deceleration	Maximální povolené zpomalení [unit/s ²]	Double (F64)
Jerk	Maximální povolená změna zrychlení [unit/s ³]	Double (F64)
Direction	Směr pohybu (jen pro cyklické osy nebo speciální případy)	Long (I32)
	1 kladný	
	2 nejkratší	
	3 záporný	
	4 aktuální	

BufferMode	Režim převzetí osy	Long (I32)
1	Aborting (nový blok se spustí okamžitě)	
2	Buffered (nový blok se spustí po dokončení předchozího)	
3	Blending low (nový blok se spustí po dokončení předchozího, původní pohyb skončí s nižší rychlostí z obou bloků)	
4	Blending high (nový blok se spustí po dokončení předchozího, původní pohyb skončí s vyšší rychlostí z obou bloků)	
5	Blending previous (nový blok se spustí po dokončení předchozího, původní pohyb skončí se svojí koncovou rychlostí)	
6	Blending next (nový blok se spustí po dokončení předchozího, původní pohyb skončí s počáteční rychlostí nového bloku)	

Výstupy

yAxis	Odkaz na osu (přípustné je jen spojení RM_Axis.axisRef-uAxis nebo yAxis-uAxis)	Reference
InVelocity	Příznak dosažení požadované rychlosti	Bool
CommandAborted	Příznak přerušení funkce bloku	Bool
Busy	Příznak, že algoritmus ještě neskončil	Bool
Active	Příznak, že blok řídí osu	Bool
Error	Příznak chyby	Bool
ErrorID	Výsledek poslední operace	Error
i	obecná chyba systému REXYGEN	

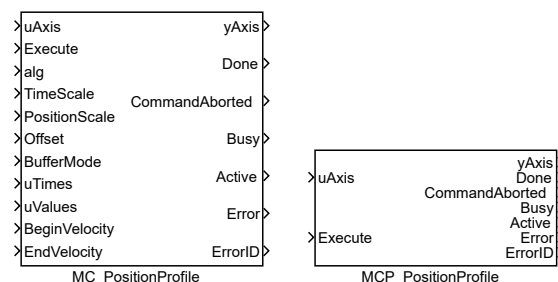
Příklad



MC_PositionProfile, MCP_PositionProfile – Generování trajektorie (poloha)

Symboly bloků

Licence: [MOTION CONTROL](#)



Popis funkce

Bloky MC_PositionProfile a MCP_PositionProfile mají naprosto shodnou funkci, jediným rozdílem je, že MCP_ varianta bloku má méně vstupů a potřebné konstanty se zadávají jako parametry bloku.

Blok MC_PositinProfile generuje takovou trajektorii, aby poloha byla požadovaná funkce času. Existují dvě možnosti, jak tuto funkci zadat:

1. tabulkou: zadávají se dvojice čísel čas a poloha. Mezi jednotlivými časy se poloha interpoluje polynomem třetího řádu (lineární interpolace není v tomto případě vhodná, protože na okrajích intervalu by byl skok v rychlosti). Hodnoty času (v sekundách) se zadávají do pole/parametru **times**, příslušné hodnoty polohy do pole/parametru **values**. Posloupnost časových okamžiků musí být stoupající a musí začínat od 0 (resp. může začínat i zápornými hodnotami, ale profil se vykonává od času 0).

2. polynomy: celá funkce se v časové ose rozdělí na několik intervalů a pro každý interval se zadá aproximující polynom pátého řádu. Časové intervaly se definují jako v předchozím případě v poli **times**. Polynom pro každý interval je ve tvaru $p(x) = a_5x^5 + a_4x^4 + a_3x^3 + a_2x^2 + a_1x + a_0$, přičemž na začátku časového intervalu je $x = 0$, a na konci $x = 1$. Koeficienty a_i jsou uloženy v poli **values** ve vzestupném pořadí (tj. pole **values** obsahuje 6 hodnot pro každý časový interval). Tato metoda umožňuje snížit počet intervalů a pro určení koeficientů polynomů existuje speciální grafický editor.

Pro obě varianty je možné zvolit rozdělení na stejně dlouhé intervaly. pak je v poli **times** jen počáteční (obvykle 0) a koncový čas.

Poznámka 1: Parametr **values** musí být ve všech případech vektor - nesmí to být matice, tj. jednotlivé hodnoty nesmí být odděleny středníkem (lze použít mezeru nebo čárku).

Poznámka 2: V režimu zadání funkce polynomem je pátého řádu a nelze to nijak měnit. Polynomy na sebe musí hladce navazovat, jinak dochází ke skokům v rychlosti a/nebo poloze. Vzhledem ke komplikovaným výpočtům je doporučeno v tomto režimu vždy používat existující speciální grafický editor.

Poznámka 3: Block neobsahuje tzv. ramp-in mode. Pokud tedy rychlost nebo poloha osy v okamžiku spuštění profilu neodpovídá počáteční rychlosti a poloze profilu, blok skončí s chybou -707 (skok v rychlosti nebo poloze). Tomuto problému s rychlostí lze předejít, pokud se použije **BufferMode=BlendingNext**. Skok v poloze se musí řešit správně nastaveným parametrem **Offset**.

Poznámka 4: Pokud na konci profilu je nenulová rychlost, osa se pohybuje dál touto rychlostí (to je v souladu se specifikací **PLCopen**).

Poznámka 5: pokud je parametr **alg=1** nebo **alg=5** nebo **alg=9** je možné nechat pole **times** prázdné (popř. nepřipojený vstup) a pak zadávat celou vačku v poli **values**, kde první sloupec jsou časové okamžiky, druhý sloupec polohy, třetí (nepovinný) sloupec rychlosti, čtvrtý (nepovinný) sloupec zrychlení.

Vstupy

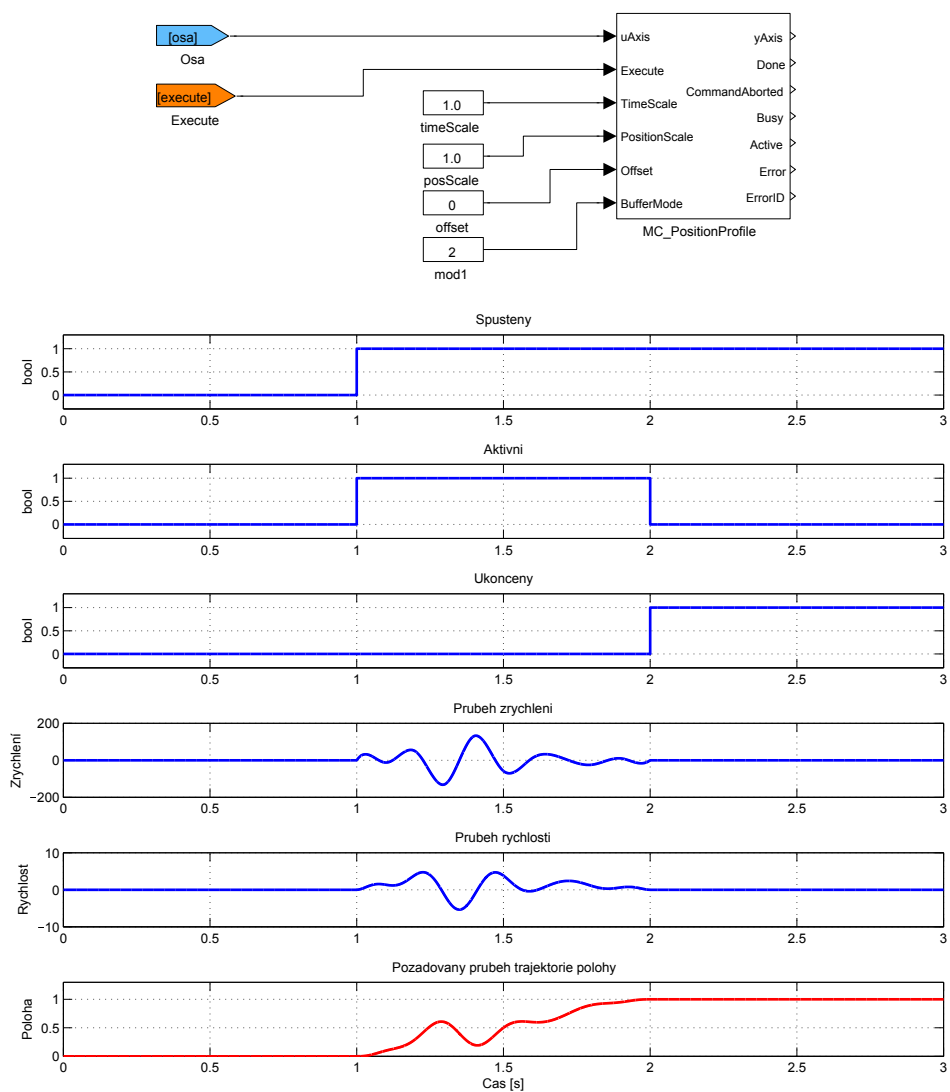
uAxis	Odkaz na osu (přípustné je jen spojení RM_Axis.axisRef-uAxis nebo yAxis-uAxis)	Reference
Execute	Náběžná hrana aktivuje blok	Bool
alg	Typ interpolace	⊙2 Long (I32)
	1 tabulka čas/hodnota (interpolace polynomem 3. řádu)	
	2 hodnoty ve stejném intervalu (interpolace polynomem 3. řádu)	
	3 interpolace polynomy 5. řádu	
	4 polynomy 5. řádu s ekvidistantními intervaly	
	5 tabulka čas/hodnota + počáteční a koncová derivace	
	6 hodnoty ve stejném intervalu + počáteční a koncová derivace	
	7 hodnoty ve stejném intervalu (aproximace B-splinem 3. řádu)	
	8 hodnoty ve stejném intervalu (aproximace B-splinem 5. řádu)	
	9 tabulka čas/hodnota (lineární interpolace)	
nmax	maximalni počet intervalů/bodů profilu	⊙3 Long (I32)
TimeScale	Konstanta násobení pro přepočet časové osy profilu	Double (F64)
PositionScale	Konstanta násobení pro přepočet hodnotové osy profilu	Double (F64)
Offset	Aditivní konstanta pro přepočet hodnotové osy profilu	Double (F64)
uTimes	vektor s časovými hodnotami	Reference
uValues	vektor s hodnotami polohy nebo koeficienty polynomů	Reference

BufferMode	Režim převzetí osy	Long (I32)
1	Aborting (nový blok se spustí okamžitě)	
2	Buffered (nový blok se spustí po dokončení předchozího)	
3	Blending low (nový blok se spustí po dokončení předchozího, původní pohyb skončí s nižší rychlostí z obou bloků)	
4	Blending high (nový blok se spustí po dokončení předchozího, původní pohyb skončí s vyšší rychlostí z obou bloků)	
5	Blending previous (nový blok se spustí po dokončení předchozího, původní pohyb skončí se svojí koncovou rychlostí)	
6	Blending next (nový blok se spustí po dokončení předchozího, původní pohyb skončí s počáteční rychlostí nového bloku)	
BeginVelocity	počáteční hodnota rychlosti (jen alg=5 nebo 6)	Double (F64)
EndVelocity	koncová hodnota rychlosti (jen alg=5 nebo 6)	Double (F64)

Výstupy

yAxis	Odkaz na osu (přípustné je jen spojení <code>RM_Axis.axisRef-uAxis</code> nebo <code>yAxis-uAxis</code>)	Reference
Done	Příznak dokončení algoritmu	Bool
CommandAborted	Příznak přerušení funkce bloku	Bool
Busy	Příznak, že algoritmus ještě neskončil	Bool
Active	Příznak, že blok řídí osu	Bool
Error	Příznak chyby	Bool
ErrorID	Výsledek poslední operace	Error
i	obecná chyba systému REXYGEN	

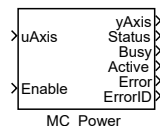
Příklad



MC_Power – Aktivace osy

Symbol bloku

Licence: [MOTION CONTROL](#)



Popis funkce

Blok **MC_Power** musí být použit s každou osou. Je to jediný blok, který převádí osu ze stavu **Disabled** do stavu **StandStill** (tj. aktivní režim). Vstup **Enable** musí být nastaven po celou dobu práce s osou. Pokud je připojena fyzická osa/měnič (block **RM_AxosSpline**), nejprve se zapne měnič a teprve potom přejde osa do stavu Standstill, popřípadě **ErrorStop** (pokud byla před vypnutím v chybě nebo pokud selhalo zapnutí měniče).

Pokud je osa vypnuta (nastavením vstupu **Enable** na nulu) při aktivním bloku (je nenulová rychlost), je nejprve aktivována zastavovací sekvence a teprve po jejím skončení je osa nastavena na stav vypnuto (disabled).

Vstupy

uAxis	Odkaz na osu (přípustné je jen spojení RM_Axis.axisRef-uAxis nebo yAxis-uAxis)	Reference
Enable	Povolení funkce bloku (aktivace výstupů)	Bool

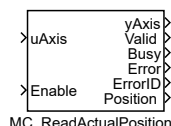
Výstupy

yAxis	Odkaz na osu (přípustné je jen spojení RM_Axis.axisRef-uAxis nebo yAxis-uAxis)	Reference
Status	Skutečný stav napájení osy	Bool
Busy	Příznak, že algoritmus ještě neskončil	Bool
Active	Příznak, že blok řídí osu	Bool
Error	Příznak chyby	Bool
ErrorID	Výsledek poslední operace i obecná chyba systému REXYGEN	Error

MC_ReadActualPosition – Skutečná poloha osy

Symbol bloku

Licence: [MOTION CONTROL](#)



Popis funkce

Blok `MC_ReadActualPosition` zpřístupňuje na výstupu `Position` aktuální polohu připojené osy. Hodnota je platná jen pokud je výstup `Valid` nenulový, čehož se dosáhne nastavením vstupu `Enable` na nenulovou hodnotu.

Blok zobrazuje logickou polohu, tj. hodnoty, které se do všech MC bloků zadávají jako poloha a která je též na výstupu `CommandedPosition` bloku [RM_Axis](#). Hodnota na snímači motoru (tj. hodnota na výstupu `PhysicalPosition` bloku [RM_Axis](#)) může být odlišná.

Poznámka: Tento blok je implementován z důvodu kompatibility s `PLCopen` neboť zobrazuje stejnou veličinu, která je přístupná i na výstupu `CommandedPosition` bloku [RM_Axis](#). Někdy se blok může hodit v rozsáhlých schématech se subsystémy.

Vstupy

<code>uAxis</code>	Odkaz na osu (přípustné je jen spojení <code>RM_Axis.axisRef-uAxis</code> nebo <code>yAxis-uAxis</code>)	Reference
<code>Enable</code>	Povolení funkce bloku (aktivace výstupů)	Bool

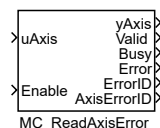
Výstupy

<code>yAxis</code>	Odkaz na osu (přípustné je jen spojení <code>RM_Axis.axisRef-uAxis</code> nebo <code>yAxis-uAxis</code>)	Reference
<code>Valid</code>	Příznak platnosti výstupní hodnoty	Bool
<code>Busy</code>	Příznak, že algoritmus ještě neskončil	Bool
<code>Error</code>	Příznak chyby	Bool
<code>ErrorID</code>	Výsledek poslední operace i obecná chyba systému REXYGEN	Error
<code>Position</code>	Aktuální poloha osy	Double (F64)

MC_ReadAxisError – Chyba osy

Symbol bloku

Licence: [MOTION CONTROL](#)



Popis funkce

Blok **MC_ReadAxisError** zpřístupňuje na výstupu **AxisErrorID** aktuální chybový kód připojené osy. Pokud osa není ve stavu chyby, hodnota tohoto výstupu je 0. Hodnota je platná jen pokud je výstup **Valid** nenulový, čehož se dosáhne nastavením vstupu **Enable** na nenulovou hodnotu.

Tento blok je implementován z důvodu kompatibility s **PLCopen** neboť zobrazuje stejnou veličinu, která je přístupná i na výstupu **ErrorID** bloku [RM_Axis](#).

Vstupy

uAxis	Odkaz na osu (přípustné je jen spojení RM_Axis.axisRef-uAxis nebo yAxis-uAxis)	Reference
Enable	Povolení funkce bloku (aktivace výstupů)	Bool

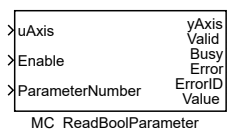
Výstupy

yAxis	Odkaz na osu (přípustné je jen spojení RM_Axis.axisRef-uAxis nebo yAxis-uAxis)	Reference
Valid	Příznak platnosti výstupní hodnoty	Bool
Busy	Příznak, že algoritmus ještě neskončil	Bool
Error	Příznak chyby	Bool
ErrorID	Výsledek poslední operace i obecná chyba systému REXYGEN	Error
AxisErrorID	Chybový kod přečtený z osy i obecná chyba systému REXYGEN	Error

MC_ReadBoolParameter – Čtení parametru (logická hodnota)

Symbol bloku

Licence: [MOTION CONTROL](#)



Popis funkce

Blok **MC_ReadBoolParameter** zpřístupňuje na výstupu **Value** aktuální hodnotu parametru připojené osy. Číslo požadovaného parametru musí být na vstupu **ParameterNumber**. Hodnota je platná jen pokud je výstup **Valid** nenulový, čehož se dosáhne nastavením vstupu **Enable** na nenulovou hodnotu.

Tento blok je implementován z důvodu kompatibility s PLCopen neboť zobrazuje parametry bloku [RM_Axis](#).

Vstupy

uAxis	Odkaz na osu (přípustné je jen spojení RM_Axis.axisRef-uAxis nebo yAxis-uAxis)	Reference
Enable	Povolení funkce bloku (aktivace výstupů)	Bool
ParameterNumber	Číslo požadovaného parametru	Long (I32)
	4 kontrola kladného omezení polohy	
	5 kontrola záporného omezení polohy	
	6 kontrola regulační odchylky (poloha)	

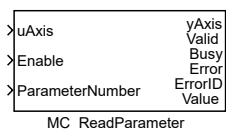
Výstupy

yAxis	Odkaz na osu (přípustné je jen spojení RM_Axis.axisRef-uAxis nebo yAxis-uAxis)	Reference
Valid	Příznak platnosti výstupní hodnoty	Bool
Busy	Příznak, že algoritmus ještě neskončil	Bool
Error	Příznak chyby	Bool
ErrorID	Výsledek poslední operace	Error
	i obecná chyba systému REXYGEN	
Value	Hodnota parametru	Bool

MC_ReadParameter – Čtení parametru (číselná hodnota)

Symbol bloku

Licence: [MOTION CONTROL](#)



Popis funkce

Blok **MC_ReadParameter** zpřístupňuje na výstupu **Value** aktuální hodnotu parametru připojené osy. Číslo požadovaného parametru musí být na vstupu **ParameterNumber**. Hodnota je platná jen pokud je výstup **Valid** nenulový, čehož se dosáhne nastavením vstupu **Enable** na nenulovou hodnotu.

Tento blok je implementován z důvodu kompatibility s **PLCopen** neboť zobrazuje parametry a výstupy bloku [RM_Axis](#).

Vstupy

uAxis	Odkaz na osu (přípustné je jen spojení RM_Axis.axisRef-uAxis nebo yAxis-uAxis)	Reference
Enable	Povolení funkce bloku (aktivace výstupů)	Bool
ParameterNumber	Číslo požadovaného parametru	Long (I32)
1	požadovaná poloha	
2	kladné omezení polohy	
3	záporné omezení polohy	
7	maximální odchylka polohy	
8	maximální rychlost (systém)	
9	maximální rychlost (bloky)	
10	skutečná rychlost	
11	požadovaná rychlost	
12	maximální zrychlení (systém)	
13	maximální zrychlení (bloky)	
14	maximální zpomalení (systém)	
15	maximální zpomalení (bloky)	
16	maximální změna zrychlení (jerk)	
1000	skutečná poloha	
1001	maximální síla/moment	
1003	skutečná síla/moment	
1004	požadovaná síla/moment	

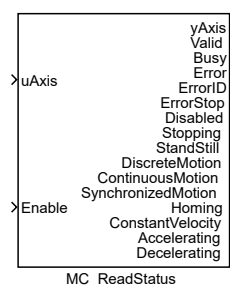
Výstupy

yAxis	Odkaz na osu (přípustné je jen spojení RM_Axis.axisRef–uAxis nebo yAxis–uAxis)	Reference
Valid	Příznak platnosti výstupní hodnoty	Bool
Busy	Příznak, že algoritmus ještě neskončil	Bool
Error	Příznak chyby	Bool
ErrorID	Výsledek poslední operace i obecná chyba systému REXYGEN	Error
Value	Hodnota parametru	Double (F64)

MC_ReadStatus – Stav osy

Symbol bloku

Licence: [MOTION CONTROL](#)



Popis funkce

Blok **MC_ReadStatus** indikuje na svých výstupech různé stavy připojené osy jako logickou hodnotu. indikovaný stav je zřejmý z názvu výstupu, popřípadě z jeho popisu. Hodnota je platná jen pokud je výstup **Valid** nenulový, čehož se dosáhne nastavením vstupu **Enable** na nenulovou hodnotu.

Vstupy

uAxis	Odkaz na osu (přípustné je jen spojení RM_Axis.axisRef-uAxis nebo yAxis-uAxis)	Reference
Enable	Povolení funkce bloku (aktivace výstupů)	Bool

Výstupy

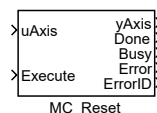
yAxis	Odkaz na osu (přípustné je jen spojení RM_Axis.axisRef-uAxis nebo yAxis-uAxis)	Reference
Valid	Příznak platnosti výstupní hodnoty	Bool
Busy	Příznak, že algoritmus ještě neskončil	Bool
Error	Příznak chyby	Bool
ErrorID	Výsledek poslední operace i obecná chyba systému REXYGEN	Error
ErrorStop	Osa je ve stavu CHYBA	Bool
Disabled	Osa je ve stavu VYPNUTO	Bool
Stopping	Osa je ve stavu STOP	Bool
StandStill	Osa je ve stavu PŘIPRAVEN	Bool
DiscreteMotion	Osa je ve stavu JEDNORÁZOVÝ POHYB	Bool
ContinuousMotion	Osa je ve stavu TRVALÝ POHYB	Bool
SynchronizedMotion	Osa je ve stavu SYNCHRONIZOVANÝ POHYB	Bool

Homing	Osa je ve stavu HLEDÁNÍ VÝCHOZÍ POLOHY	Bool
ConstantVelocity	Osa se pohybuje konstantní rychlostí	Bool
Accelerating	Osa zrychluje	Bool
Decelerating	Osa brzdí/zpomaluje	Bool

MC_Reset – Nulování chyb osy

Symbol bloku

Licence: [MOTION CONTROL](#)



Popis funkce

Blok **MC_Reset** převede připojenou osu ze stavu „ErrorStop“ do stavu „StandStill“ a vynuluje v ose všechny příznaky chyby. Je to v podstatě jediný blok, který ve stavu „ErrorStop“ nehlásí chybu.

Vstupy

uAxis	Odkaz na osu (přípustné je jen spojení RM_Axis.axisRef-uAxis nebo yAxis-uAxis)	Reference
Execute	Náběžná hrana aktivuje blok	Bool

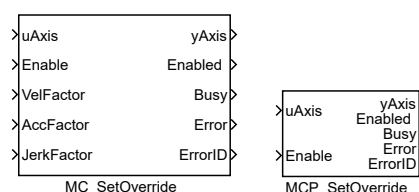
Výstupy

yAxis	Odkaz na osu (přípustné je jen spojení RM_Axis.axisRef-uAxis nebo yAxis-uAxis)	Reference
Done	Příznak dokončení algoritmu	Bool
Busy	Příznak, že algoritmus ještě neskončil	Bool
Error	Příznak chyby	Bool
ErrorID	Výsledek poslední operace i obecná chyba systému REXYGEN	Error

MC_SetOverride, MCP_SetOverride – Nastavení násobivých faktorů na ose

Symbody bloků

Licence: [MOTION CONTROL](#)



Popis funkce

Bloky MC_SetOverride a MCP_SetOverride mají naprosto shodnou funkci, jediným rozdílem je, že MCP_ varianta bloku má méně vstupů a potřebné konstanty se zadávají jako parametry bloku.

Blok **MC_SetOverride** nastavuje násobivé faktory které se projeví ve všech blocích pracujících s osou. Hodnoty rychlosti, zrychlení a jerku ve všech blocích je potřeba vynásobit faktorem z tohoto bloku, abychom dostali hodnotu, se kterou blok skutečně pracuje. Toto se netýká limitních hodnot zadaných v [RM_Axis](#) a administrativních bloků.

Tento blok není aktivován hranou, ale pokud je vstup **Enable** nenulový, tak se hodnoty trvale aktualizují. Pokud je aktivní blok typu [MC_MoveAbsolute](#), vede to na neustálé přepočítávání trajektorie, což je výpočetně (a tím i časově) náročná operace a navíc se kumulují zaokrouhlovací chyby. Proto je zavedena necitlivost (parametr **diff**) a přepočet trajektorie je proveden až když se některý z faktorů změní více, než je tato necitlivost.

Poznámka: Všechny faktory musí být kladné. Faktory větší než 1 jsou možné, ale často vedou k překročení mezí nastavených na ose a k selhání pohybu (blok hlásí chybu **errorID** = -700 - neplatný parametr) nebo dokonce k havarijnímu zastavení osy (blok pak hlásí chybu **errorID** = -701 - hodnota mimo rozsah).

Vstupy

uAxis	Odkaz na osu (přípustné je jen spojení RM_Axis.axisRef-uAxis nebo yAxis-uAxis)	Reference
Enable	Povolení funkce bloku (aktivace výstupů)	Bool
VelFactor	Faktor násobení pro rychlost	Double (F64)
AccFactor	Faktor násobení pro zrychlení	Double (F64)
JerkFactor	Faktor násobení pro změnu zrychlení	Double (F64)

Výstupy

<code>yAxis</code>	Odkaz na osu (přípustné je jen spojení <code>RM_Axis.axisRef-uAxis</code> nebo <code>yAxis-uAxis</code>)	Reference
<code>Enabled</code>	Povolení funkce bloku (aktivace výstupů)	Bool
<code>Busy</code>	Příznak, že algoritmus ještě neskončil	Bool
<code>Error</code>	Příznak chyby	Bool
<code>ErrorID</code>	Výsledek poslední operace i obecná chyba systému REXYGEN	Error

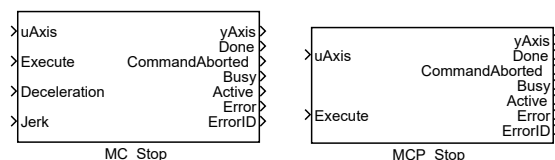
Parametr

<code>diff</code>	Pásmo necitlivosti (pro přepočet trajektorie) ↓0.0 ↑1.0 ⊙0.1	Double (F64)

MC_Stop, MCP_Stop – Zastavení pohybu

Symboly bloků

Licence: [MOTION CONTROL](#)



Popis funkce

Bloky MC_Stop a MCP_Stop mají naprosto shodnou funkci, jediným rozdílem je, že MCP_ varianta bloku má méně vstupů a potřebné konstanty se zadávají jako parametry bloku.

Block MC_Stop provede zastavovací sekvenci a převede osu do stavu **Stopping**. V tomto stav není možné spustit žádný pohyb a osa v něm zůstává dokud je vstup **Execute** nenulový.

Poznámka1: Blok nemá parametr **BufferMode**. Mód je vždy **Aborting**.

Poznámka2: Protože selhání příkazu k zastavení může být nebezpečné, blok generuje chybu jen v naprosto fatálních případech (např. nezapojený vstup **uAxis**) a snaží se co nejkorektněji zastavit (např. při nekorektních parametrech použije nastavení osy nebo vyvolá sekvenci pro chybové zastavení).

Vstupy

uAxis	Odkaz na osu (přípustné je jen spojení RM_Axis.axisRef-uAxis nebo yAxis-uAxis)	Reference
Execute	Náběžná hrana aktivuje blok	Bool
Deceleration	Maximální povolené zpomalení [unit/s ²]	Double (F64)
Jerk	Maximální povolená změna zrychlení [unit/s ³]	Double (F64)

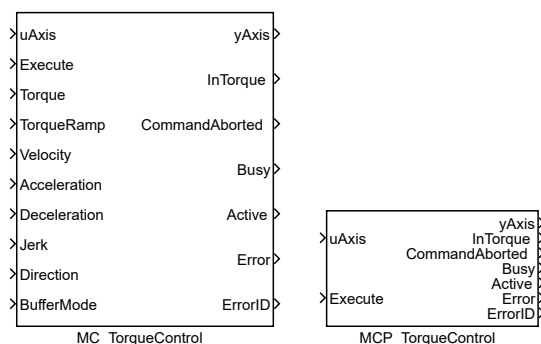
Výstupy

yAxis	Odkaz na osu (přípustné je jen spojení RM_Axis.axisRef-uAxis nebo yAxis-uAxis)	Reference
Done	Příznak dokončení algoritmu	Bool
CommandAborted	Příznak přerušení funkce bloku	Bool
Busy	Příznak, že algoritmus ještě neskončil	Bool
Active	Příznak, že blok řídí osu	Bool
Error	Příznak chyby	Bool

ErrorID	Výsledek poslední operace	Error
i	obecná chyba systému REXYGEN	

MC_TorqueControl, MCP_TorqueControl – Řízení síly/momentu

Symboly bloků

Licence: [MOTION CONTROL](#)

Popis funkce

Bloky MC_TorqueControl a MCP_TorqueControl mají naprosto shodnou funkci, jediným rozdílem je, že MCP_ varianta bloku má méně vstupů a potřebné konstanty se zadávají jako parametry bloku.

Blok MC_TorqueControl generuje požadovaný moment/sílu nejprve s konstantním nárůstem (parametr TorqueRamp) a po dosažení maximální hodnoty (parametr Torque) je již moment/síla konstantní. Pohyb osy je řízen podle požadovaného momentu tak, aby nebyly překročeny maximální hodnoty rychlosti, zrychlení/zpomalení a případně jerku.

Vstupy

uAxis	Odkaz na osu (přípustné je jen spojení RM_Axis.axisRef-uAxis nebo yAxis-uAxis)	Reference
Execute	Náběžná hrana aktivuje blok	Bool
Torque	Maximální povolený moment/síla	Double (F64)
TorqueRamp	Maximální povolená změna momentu/síly	Double (F64)
Velocity	Maximální povolená rychlost [unit/s]	Double (F64)
Acceleration	Maximální povolené zrychlení [unit/s ²]	Double (F64)
Deceleration	Maximální povolené zpomalení [uunit/s ²]	Double (F64)
Jerk	Maximální povolená změna zrychlení [unit/s ³]	Double (F64)
Direction	Směr pohybu (jen pro cyklické osy nebo speciální případy)	Long (I32)
	1 kladný	
	2 nejkratší	
	3 záporný	
	4 aktuální	

BufferMode	Režim převzetí osy	Long (I32)
1	Aborting (nový blok se spustí okamžitě)	
2	Buffered (nový blok se spustí po dokončení předchozího)	
3	Blending low (nový blok se spustí po dokončení předchozího, původní pohyb skončí s nižší rychlostí z obou bloků)	
4	Blending high (nový blok se spustí po dokončení předchozího, původní pohyb skončí s vyšší rychlostí z obou bloků)	
5	Blending previous (nový blok se spustí po dokončení předchozího, původní pohyb skončí se svojí koncovou rychlostí)	
6	Blending next (nový blok se spustí po dokončení předchozího, původní pohyb skončí s počáteční rychlostí nového bloku)	

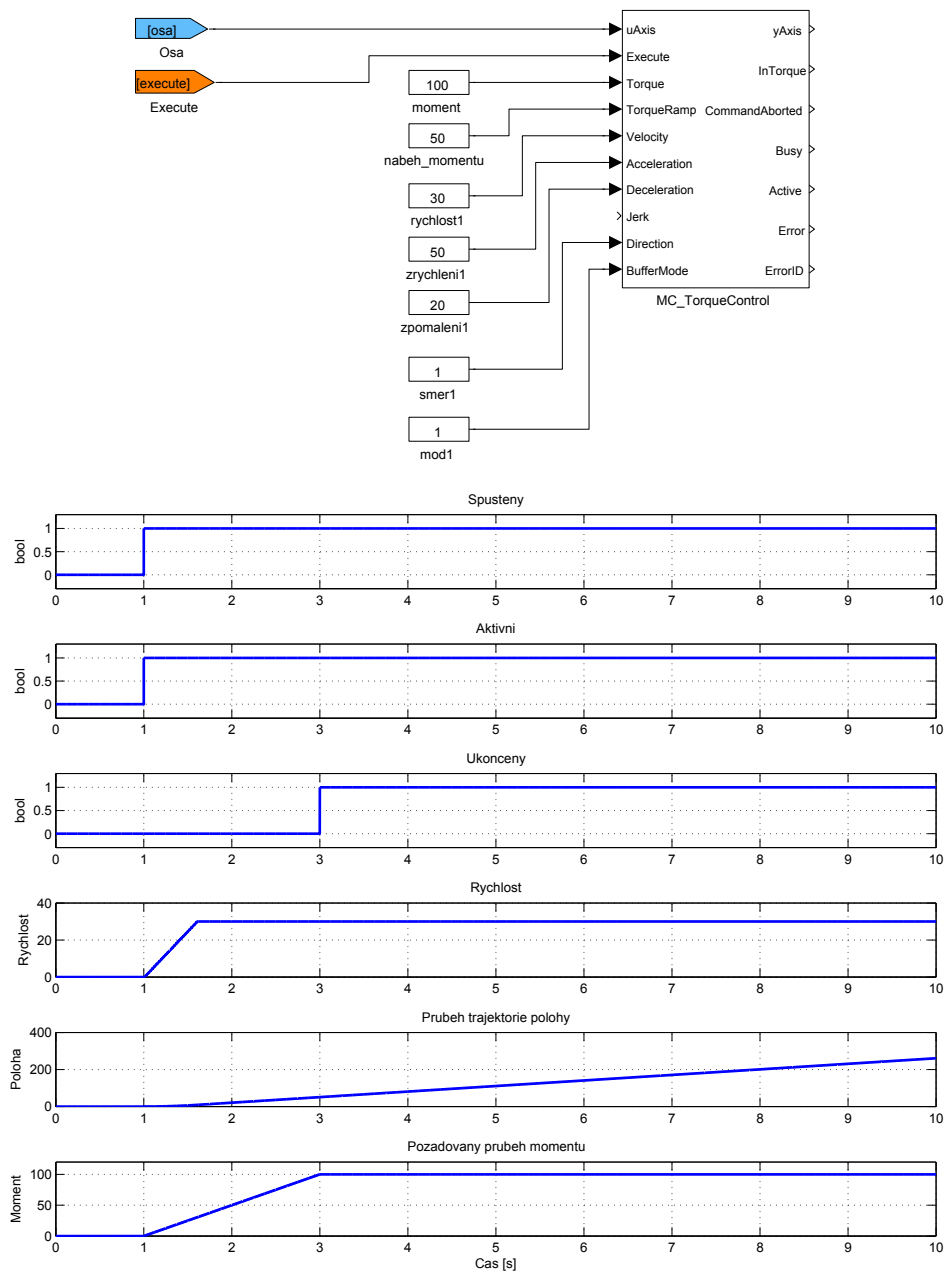
Výstupy

yAxis	Odkaz na osu (přípustné je jen spojení <code>RM_Axis.axisRef-uAxis</code> nebo <code>yAxis-uAxis</code>)	Reference
InTorque	Příznak dosažení požadovaného momentu/síly	Bool
CommandAborted	Příznak přerušení funkce bloku	Bool
Busy	Příznak, že algoritmus ještě neskončil	Bool
Active	Příznak, že blok řídí osu	Bool
Error	Příznak chyby	Bool
ErrorID	Výsledek poslední operace	Error
i	obecná chyba systému REXYGEN	

Parametr

kma	Poměr mezi silou/momentem a zrychlením	Double (F64)

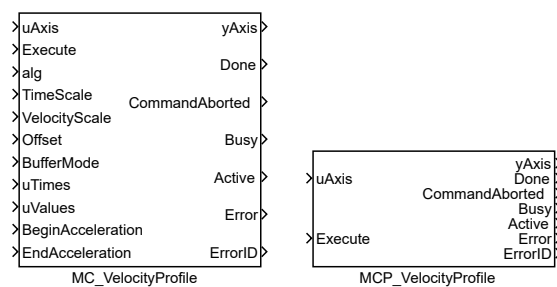
Příklad



MC_VelocityProfile, MCP_VelocityProfile – Generování trajektorie (rychlost)

Symbody bloků

Licence: [MOTION CONTROL](#)



Popis funkce

Bloky MC_VelocityProfile a MCP_VelocityProfile mají naprosto shodnou funkci, jediným rozdílem je, že MCP_ varianta bloku má méně vstupů a potřebné konstanty se zadávají jako parametry bloku.

Blok **MC_VelocityProfile** generuje takovou trajektorii, aby rychlost byla požadovaná funkcí času. Existují dvě možnosti, jak tuto funkci zadat:

1. tabulkou: zadávají se dvojice čísel čas a rychlost. Mezi jednotlivými časy se hodnota rychlosti interpoluje lineárně. Hodnoty času (v sekundách) se zadávají do pole/parametru **times**, příslušné hodnoty rychlosti do pole/parametru **values**. Posloupnost časových okamžiků musí být stoupající a musí začínat od 0 (resp. může začínat i zápornými hodnotami, ale profil se vykonává od času 0).

2. polynomy: celá funkce se v časové ose rozdělí na několik intervalů a pro každý interval se zadá aproximující polynom pátého řádu. Časové intervaly se definují jako v předchozím případě v poli **times**. Polynom pro každý interval je ve tvaru $p(x) = a_5x^5 + a_4x^4 + a_3x^3 + a_2x^2 + a_1x + a_0$, přičemž na začátku časového intervalu je $x = 0$, a na konci $x = 1$. Koefficienty a_i jsou uloženy v poli **values** ve vzestupném pořadí (tj. pole **values** obsahuje 6 hodnot pro každý časový interval). Tato metoda umožňuje snížit počet intervalů a pro určení koeficientů polynomů existuje speciální grafický editor.

Pro obě varianty je možné zvolit rozdělení na stejně dlouhé intervaly. pak je v poli **uTimes** jen počáteční (obvykle 0) a koncový čas.

Poznámka 1: Vstup/parametr **uValues** musí být ve všech případech vektor - nesmí to být matice, tj. jednotlivé hodnoty nesmí být odděleny středníkem (lze použít mezeru nebo čárku).

Poznámka 2: Dialog je jednotný pro **MC_AccelerationProfile**, **MC_VelocityProfile**, **MC_PositionProfile**. Některé režimy (parametr **alg** nedávají pro **VelocityProfile** smysl

a nefungují (aproximace B-spline nefunguje, režimy 1,2,5,6 používají lineární interpolaci)

Poznámka 3: V režimu zadání funkce polynomem je hodnota polynomu poloha a polynom je vždy pátého řádu a nelze to nijak měnit. **VelocityScale** a **Offset** je samozřejmě pro rychlost. Vzhledem ke komplikovaným výpočtům je doporučeno v tomto režimu vždy používat existující speciální grafický editor.

Poznámka 4: Block neobsahuje tzv. ramp-in mode. Pokud tedy rychlost osy v okamžiku spuštění profilu neodpovídá počáteční rychlosti profilu, blok skončí s chybou -707 (skok v rychlosti nebo poloze). Tomuto problému lze předejít, pokud se použije **BufferMode=BlendingNext** nebo je potřeba správně nastavit parametr **Offset**.

Poznámka6: Pokud na konci profilu je nenulová rychlost, osa se pohybuje dál touto rychlostí (to je v souladu se specifikací **PLCopen**).

Vstupy

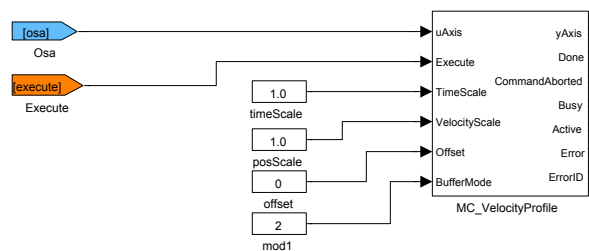
uAxis	Odkaz na osu (přípustné je jen spojení RM_Axis.axisRef-uAxis nebo yAxis-uAxis)	Reference
Execute	Náběžná hrana aktivuje blok	Bool
alg	Typ interpolace	⊙2 Long (I32)
	1 tabulka čas/hodnota (interpolace polynomem 3. řádu)	
	2 hodnoty ve stejném intervalu (interpolace polynomem 3. řádu)	
	3 interpolace polynomy 5. řádu	
	4 polynomy 5. řádu s ekvidistantními intervaly	
	5 tabulka čas/hodnota + počáteční a koncová derivace	
	6 hodnoty ve stejném intervalu + počáteční a koncová derivace	
	7 hodnoty ve stejném intervalu (aproximace B-splinem 3. řádu)	
	8 hodnoty ve stejném intervalu (aproximace B-splinem 5. řádu)	
	9 tabulka čas/hodnota (lineární interpolace)	
nmax	maximalni počet intervalů/bodů profilu	⊙3 Long (I32)
TimeScale	Konstanta násobení pro přepočet časové osy profilu	Double (F64)
VelocityScale	Konstanta násobení pro přepočet hodnotové osy profilu	Double (F64)
Offset	Aditivní konstanta pro přepočet hodnotové osy profilu	Double (F64)
BufferMode	Režim převzetí osy	Long (I32)
uTimes	vektor s časovými hodnotami	Reference

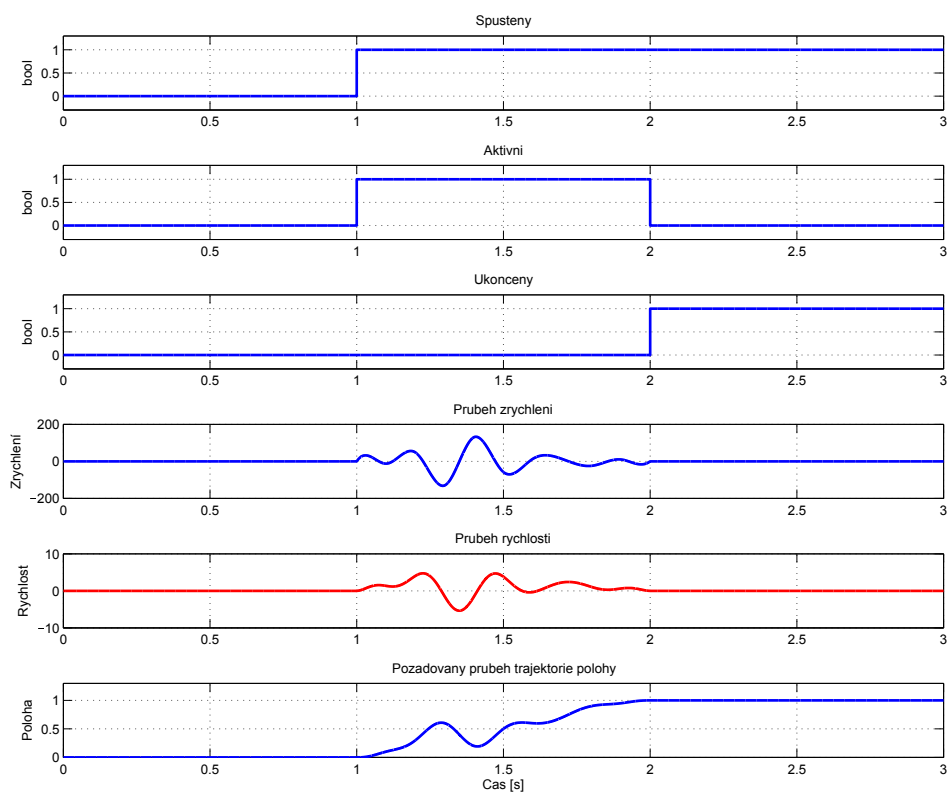
uValues	vektor s hodnotami rychlosti nebo koeficienty polynomů	Reference
1	Aborting (nový blok se spustí okamžitě)	
2	Buffered (nový blok se spustí po dokončení předchozího)	
3	Blending low (nový blok se spustí po dokončení předchozího, původní pohyb skončí s nižší rychlostí z obou bloků)	
4	Blending high (nový blok se spustí po dokončení předchozího, původní pohyb skončí s vyšší rychlostí z obou bloků)	
5	Blending previous (nový blok se spustí po dokončení předchozího, původní pohyb skončí se svojí koncovou rychlostí)	
6	Blending next (nový blok se spustí po dokončení předchozího, původní pohyb skončí s počáteční rychlostí nového bloku)	
BeginAcceleration	počáteční hodnota zrychlení (jen alg=5 nebo 6)	Double (F64)
EndAcceleration	koncová hodnota (jen alg=5 nebo 6)	Double (F64)

Výstupy

yAxis	Odkaz na osu (přípustné je jen spojení <code>RM_Axis.axisRef-uAxis</code> nebo <code>yAxis-uAxis</code>)	Reference
Done	Příznak dokončení algoritmu	Bool
CommandAborted	Příznak přerušení funkce bloku	Bool
Busy	Příznak, že algoritmus ještě neskončil	Bool
Active	Příznak, že blok řídí osu	Bool
Error	Příznak chyby	Bool
ErrorID	Výsledek poslední operace	Error
i	obecná chyba systému REXYGEN	

Příklad

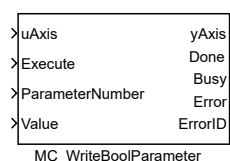




MC_WriteBoolParameter – Nastavení parametru (logická hodnota)

Symbol bloku

Licence: [MOTION CONTROL](#)



Popis funkce

Blok `MC_WriteBoolParameter` změní hodnotu parametru připojené osy na hodnotu danou vstupem `Value`. Číslo požadovaného parametru musí být přivedeno na vstup `ParameterNumber`.

Tento blok je implementován z důvodu kompatibility s `PLCopen` neboť nastavuje parametry bloku [RM_Axis](#), což umožňuje i blok `SETPB`.

Vstupy

<code>uAxis</code>	Odkaz na osu (přípustné je jen spojení <code>RM_Axis.axisRef-uAxis</code> nebo <code>yAxis-uAxis</code>)	Reference
<code>Execute</code>	Náběžná hrana aktivuje blok	Bool
<code>ParameterNumber</code>	Číslo požadovaného parametru	Long (I32)
	4 kontrola kladného omezení polohy	
	5 kontrola záporného omezení polohy	
	6 kontrola regulační odchylky (poloha)	
<code>Value</code>	Hodnota parametru	Bool

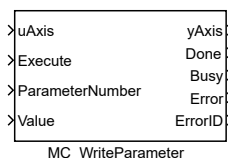
Výstupy

<code>yAxis</code>	Odkaz na osu (přípustné je jen spojení <code>RM_Axis.axisRef-uAxis</code> nebo <code>yAxis-uAxis</code>)	Reference
<code>Done</code>	Příznak dokončení algoritmu	Bool
<code>Busy</code>	Příznak, že algoritmus ještě neskončil	Bool
<code>Error</code>	Příznak chyby	Bool
<code>ErrorID</code>	Výsledek poslední operace	Error
	i obecná chyba systému REXYGEN	

MC_WriteParameter – Nastavení parametru (číselná hodnota)

Symbol bloku

Licence: [MOTION CONTROL](#)



Popis funkce

Blok **MC_WriteParameter** změní hodnotu parametru připojené osy na hodnotu danou vstupem **Value**. Číslo požadovaného parametru musí být na vstupu **ParameterNumber**.

Tento blok je implementován z důvodu kompatibility s **PLCopen** neboť nastavuje parametry bloku [RM_Axis](#), což umožňuje i blok **SETPR**.

Vstupy

uAxis	Odkaz na osu (přípustné je jen spojení RM_Axis.axisRef–uAxis nebo yAxis–uAxis)	Reference
Execute	Náběžná hrana aktivuje blok	Bool
ParameterNumber	Číslo požadovaného parametru	Long (I32)
	2 kladné omezení polohy	
	3 záporné omezení polohy	
	7 maximální odchylka polohy	
	8 maximální rychlost (systém)	
	9 maximální rychlost (bloky)	
	13 maximální zrychlení (bloky)	
	15 maximální zpomalení (bloky)	
	16 maximální změna zrychlení (jerk)	
	1001 .. maximální síla/moment	
Value	Hodnota parametru	Double (F64)

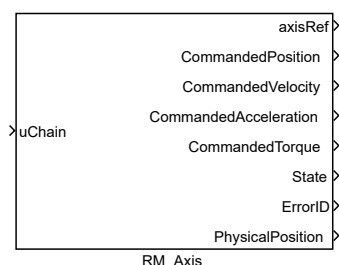
Výstupy

yAxis	Odkaz na osu (přípustné je jen spojení RM_Axis.axisRef–uAxis nebo yAxis–uAxis)	Reference
Done	Příznak dokončení algoritmu	Bool
Busy	Příznak, že algoritmus ještě neskončil	Bool
Error	Příznak chyby	Bool
ErrorID	Výsledek poslední operace	Error
	i obecná chyba systému REXYGEN	

RM_Axis – Osa pro řízení pohybu

Symbol bloku

Licence: [MOTION CONTROL](#)



Popis funkce

Blok **RM_AXIS** je základní blok osy pro řízení pohybu. Představuje sdílenou strukturu, kde jsou uloženy všechny stavy a parametry osy. Algoritmus tohoto bloku se stará o kontrolu nastavených mezí, havarijní zastavení v případě potřeby a přepočty všech stavů a výstupů pro případ, že žádný blok není aktivní, ale osa (motor) má nenulovou rychlost. Výstupem tohoto bloku jsou pouze požadované hodnoty polohy, rychlosti, popřípadě zrychlení a momentu. Pokud nejde o virtuální osu a osa je spojena s reálným motorem, je nutné realizovat ještě regulátor polohy a rychlosti, který není součástí tohoto bloku – zpětnovazební signály jsou použity pouze pro kontrolu správné činnosti regulátoru, například odchylka nesmí překročit stanovenou mez, viz dále. Dále je mohou využívat některé speciální bloky, například bloky pro nalezení výchozí polohy. Zpětnovazební signál se připojuje pomocí bloku [RM_AxisSpline](#).

Parametry tohoto bloku jsou stejné, jaké vyžaduje **PLCopen** pro osu. Pokud jsou zadány parametry nesprávně (nekonzistentně), výstup **errorID** je nastaven na hodnotu -700 (neplatný parametr) a všechny ostatní bloky navázané na osu skončí s chybou -703 (chybný stav).

Parametry pro omezení rychlosti a zrychlení jsou dvojí. Aplikační parametry omezují hodnoty, které lze zadat do pohybových bloků, ale skutečná rychlost nebo zrychlení může být někdy větší. Systémové parametry se překročit nesmí a pokud k tomu dojde, je pohyb zastaven a osa přejde do stavu errorstop.

Implicitní hodnoty parametrů (zejména limity na rychlost a zrychlení) jsou záměrně nastaveny na 0, což je nedovolená hodnota. Všechny parametry tak musí nastavit uživatel podle skutečných možností připojeného motoru a stroje.

Vstupy

uChain	blok vstup nepoužívá, ale je možné připojit libovolný signál a tím definovat pořadí spouštění bloků	Long (I32)

Výstupy

axisRef	Odkaz na osu (přípustné je jen spojení RM_Axis.axisRef-uAxis nebo yAxis-uAxis)	Reference
CommandedPosition	Požadovaná poloha osy Poloha, která se zadává do pohybových bloků. V případě homingu nebo cyklické osy se liší od PhysicalPosition	Double (F64)
CommandedVelocity	Požadovaná rychlost osy	Double (F64)
CommandedAcceleration	Požadované zrychlení osy	Double (F64)
CommandedTorque	Požadovaný moment/síla	Double (F64)
State	Stav osy pro řízení pohybu	Long (I32)
	0 Disabled (osa blokována)	
	1 Stand still (osa připravena)	
	2 Homing (hledání výchozí polohy)	
	3 Discrete motion (jednorázový pohyb)	
	4 Continuous motion (trvalý pohyb)	
	5 Synchronized motion (pohyb synchronní s jinou osou)	
	6 Coordinated motion (osa řízena blokem pro koordinovaný pohyb)	
	7 Stopping (zastavování nebo dočasné blokování)	
	8 Error stop (zastavování nebo blokování osy po chybě)	
	9 Drive error (jako errorstop, ale chyba je způsobena vnějším signálem - obvykle chyba měniče)	
ErrorID	Kod chyby ve stavu errorstop	Error
	i obecná chyba systému REXYGEN	
PhysicalPosition	Požadovaná poloha osy Poloha, která se zapojuje do zpětnovazebního regulátoru. V případě homingu nebo cyklické osy se liší od CommandedPosition	Double (F64)

Parametry

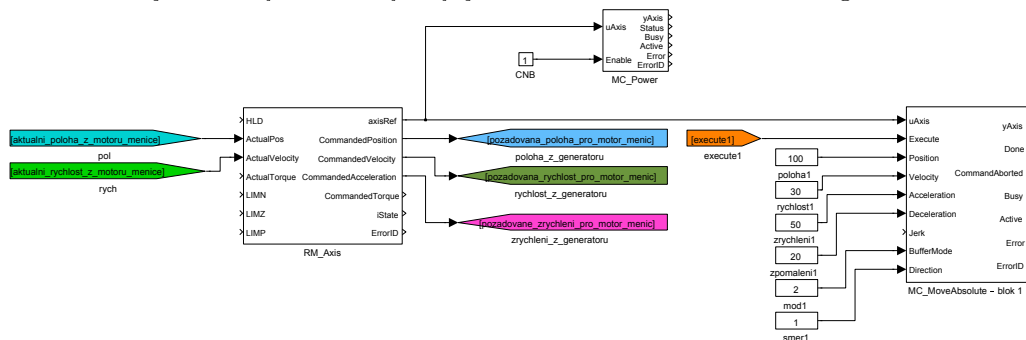
AxisType	Druh osy	⊙1 Long (I32)
	1 lineární osa	
	2 cyklická osa s cyklickým polohovým senzorem	
	3 cyklická osa s lineárním polohovým senzorem	
	4 lineární osa v testovacím režimu. V tomto režimu jsou vypnuté některé kontroly, zejména polohové limity (včetně koncových spínačů). Tento typ je určen pro pokusy ožívování, někdy se může hodit i pro trvalý provoz na cyklických osách.	
EnableLimitPos	Příznak kontroly maximální polohy v kladném směru (tj. když je zaškrtnuto, je platné MaxPosAppl)	Bool
MaxPosAppl	Maximální hodnota v kladném směru pro pohybové bloky. Hodnota musí být menší než MaxPosSystem pro lineární osu. Pro cyklickou osu s lineárním senzorem hodnota omezuje na několik otáček (vhodné pro robotické aplikace) a musí být větší než MaxPosSystem .	Double (F64)

MaxPosSystem	Maximální hodnota v kladném směru (fyzický doraz)	Double (F64)
EnableLimitNeg	Příznak kontroly maximální polohy v záporném směru (tj. když je zaškrtnuto, je platné MinPosAppl)	Bool
MinPosAppl	Maximální hodnota v záporném směru pro pohybové bloky Hodnota musí být větší než (nastává dříve) MinPosSystem pro lineární osu. Pro cyklickou osu s lineárním senzorem hodnota omezuje na několik otáček (vhodné pro robotické aplikace) a musí být menší než (nastává později) MaxPosSystem .	Double (F64)
MinPosSystem	Maximální hodnota v záporném směru (fyzický doraz)	Double (F64)
EnablePosLagMonitor	Příznak kontroly skluzu (rozdíl mezi požadovanou a skutečnou polohou)	Bool
MaxPositionLag	Maximální hodnota skluzu	Double (F64)
MaxVelocitySystem	Maximální hodnota rychlosti (havarijní)	Double (F64)
MaxVelocityAppl	Maximální hodnota rychlosti (pracovní)	Double (F64)
MaxAccelerationSystem	Maximální hodnota zrychlení (havarijní)	Double (F64)
MaxAccelerationAppl	Maximální hodnota zrychlení (pracovní)	Double (F64)
MaxDecelerationSystem	Maximální hodnota brždění (havarijní)	Double (F64)
MaxDecelerationAppl	Maximální hodnota brždění (pracovní)	Double (F64)
DefaultJerk	Maximální doporučená změna zrychlení [unit/s ³] Skutečná změna zrychlení se nekontroluje a může překročit tuto hodnotu.	Double (F64)
MaxTorque	Maximální hodnota momentu/síly	Double (F64)
TorqueRatio	Poměr momentu a zrychlení. Pokud moment osy není určen jiným způsobem (např. pomocí bloku MC_TorqueControl), určuje se moment ze zrychlení pomocí tohoto koeficientu.	Double (F64)
LoopDelay	zpoždění mezi požadovanou a skutečnou hodnotou [s] Zpoždění nastává vlivem periody vzorkování, komunikace s externím kontrolérem, zpožděním regulační smyčky. Nastavení umožňuje přesněji kontrolovat PositionLag . (not yet implemented)	Double (F64)
StartMode	Volby režimu při zapnutí 1 při zapnutí je pohyb zastaven 2 při zapnutí se pohyb pokračuje aktuální rychlostí motoru	⊙1 Long (I32)
HomingRequired	Pokud je zaškrtnuto, tak jsou blokovány všechny pohyby (kromě MC_Home), dokud se neprovede homing	Bool

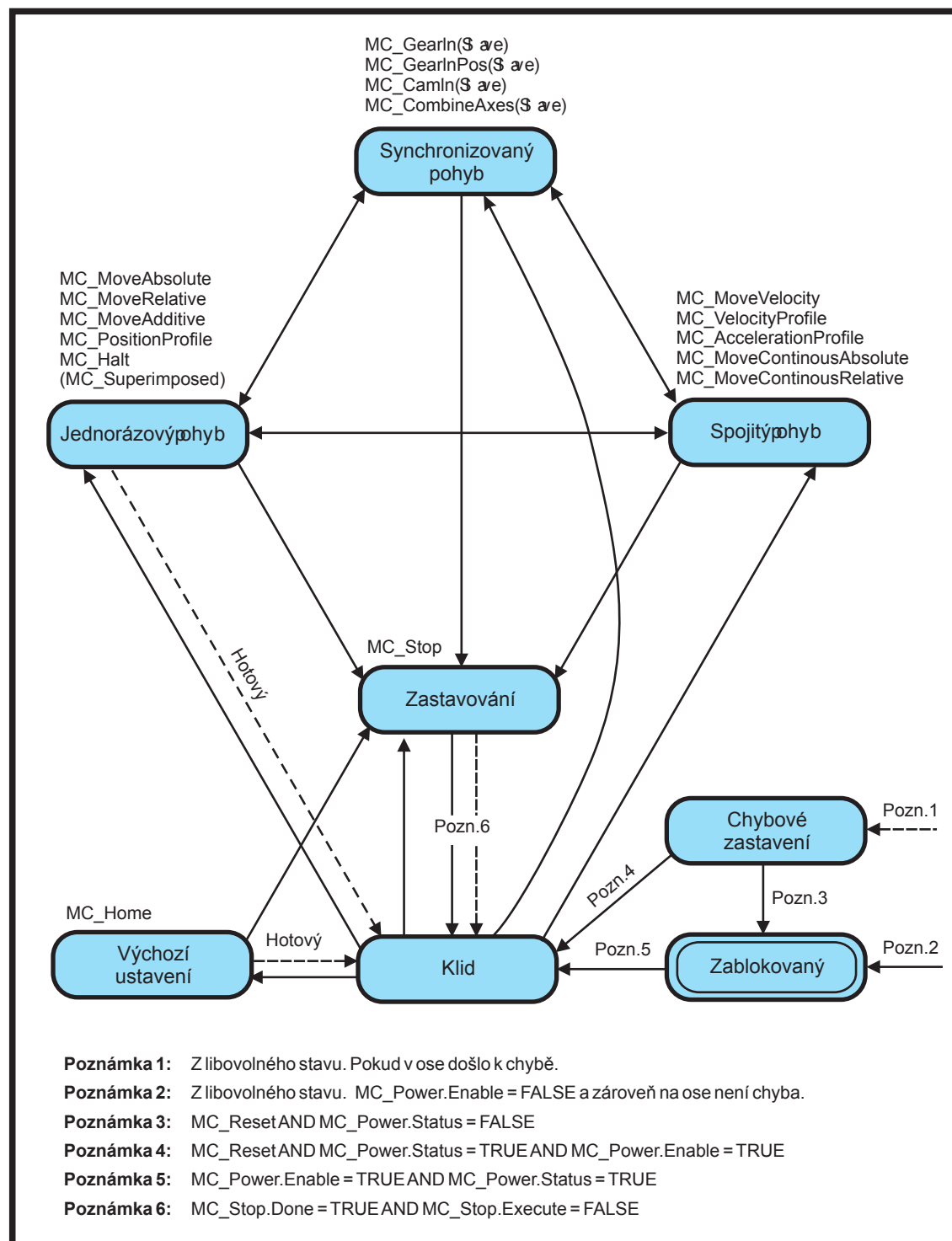
Příklad

Následující schéma znázorňuje základní zapojení bloků z knihovny Motion Control v minimální realizaci nutné pro zahájení pohybu. Fyzická osa je reprezentována blokem **RM_Axis** a jí příslušnou datovou strukturou. Po nastavení omezení na požadovaný průběh pohybu lze na vstupy bloku připojit signál aktuální polohy, rychlosti a momentu (využíváno pro kontrolu skluzu) popřípadě logické signály koncových spínačů pro nalezení referenční výchozí polohy. Výstupní signál **axisRef** se připojuje na vstup všech bloků pracujících s danou osou. Před zahájením pohybu je třeba osu aktivovat blokem **MC_Power**, čímž dojde k přechodu ze zablokovaného do klidového stavu. V tuto chvíli

je již možné zahájit libovolný jednorázový, spojitý nebo synchronizovaný pohyb spuštěním příslušného funkčního bloku. Generovaná trajektorie pohybu, tedy požadovaný průběh polohy, rychlosti a zrychlení je k dispozici na výstupech bloku **RM_Axis**. Odtud může být přiveden do regulační struktury pohonu, která existuje buďto lokálně v systému REXYGEN ve stejném nebo jiném tasku nebo je implementována přímo v řídicí jednotce motoru (zesilovač, frekvenční měnič), kam může být ze systému REXYGEN předána s použitím některé ze standardních sériových komunikací. V případě zadání chybných parametrů, volání nepřipustného příkazu nebo při překročení maximální hodnoty skluzu dochází k nouzovému zastavení, osa přechází do režimu chyby, kterou indikuje na výstupu **ErrorID**. Pro opětovné zahájení pohybu je třeba chybu kvitovat blokem **MC_Reset**. Přechod mezi jednotlivými režimy osy je znázorněn na stavovém diagramu.



Stavový diagram osy

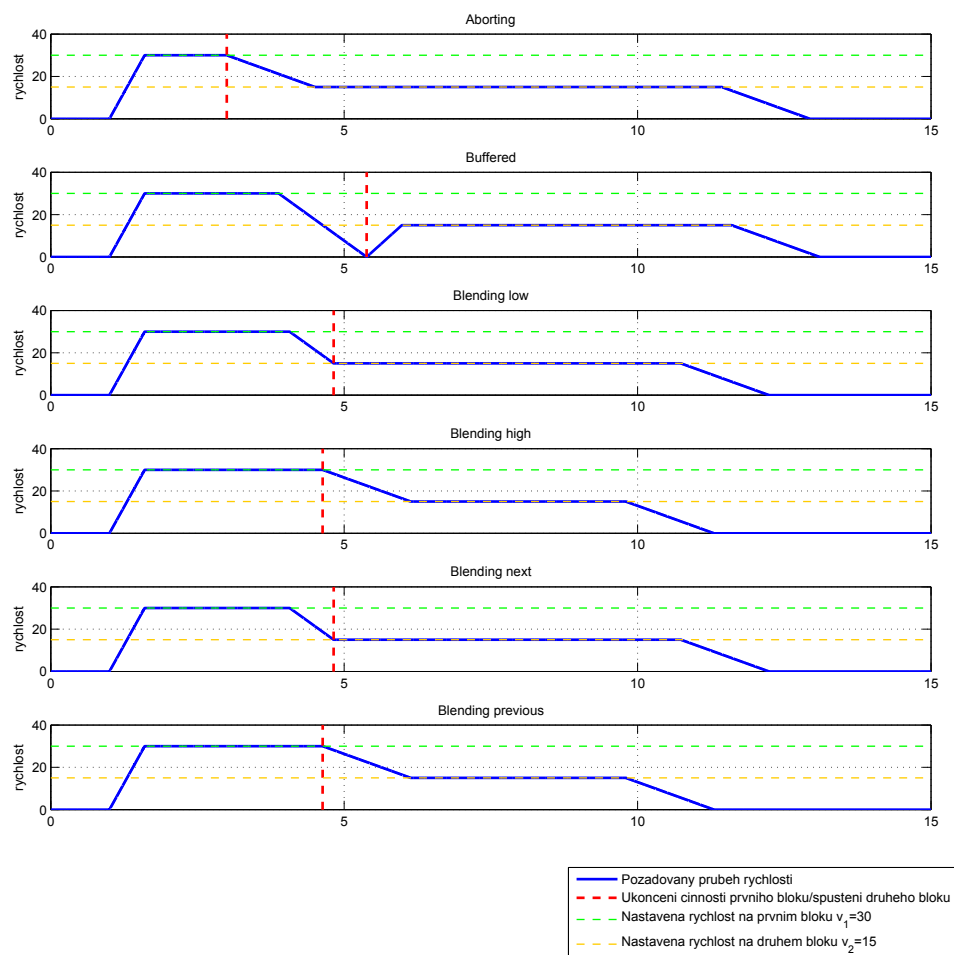


Míchání pohybů

Norma PLCOpen definuje pro bloky spouštějící pohyb osy vstupní parametr **BufferMode**, který určuje chování osy v případě, že je za běhu zavolán nový příkaz ve formě jiného funkčního bloku. Tento přechod mezi různými pohyby (míchání, "Blending") lze řešit několika způsoby. Následující tabulka podává stručné vysvětlení funkce jednotlivých režimů míchání pohybu a jejich vliv na tvar výsledné generované trajektorie. Detailní popis lze nalézt ve specifikaci PLCOpen.

Aborting	nový pohyb okamžitě přerušuje původní příkaz
Buffered	nový pohyb je vykonán po dokončení původního příkazu, neprobíhá žádné míchání
Blending low	nový pohyb je vykonán po dokončení původního příkazu, osa však nezastavuje v původní cílové pozici, ale končí zde rychlostí určenou jako nejnižší hodnota ze dvou limitů pro maximální rychlost zadanou na vstupu dvou bloků, které přebírají řízení osy
Blending high	nový pohyb je vykonán po dokončení původního příkazu, osa však nezastavuje v původní cílové pozici, ale končí zde rychlostí určenou jako nejvyšší hodnota ze dvou limitů pro maximální rychlost zadanou na vstupu dvou bloků, které přebírají řízení osy
Blending previous	nový pohyb je vykonán po dokončení původního příkazu, osa však nezastavuje v původní cílové pozici, ale končí zde rychlostí danou limitem pro maximální rychlost na vstupu prvního bloku
Blending next	nový pohyb je vykonán po dokončení původního příkazu, osa však nezastavuje v původní cílové pozici, ale končí zde rychlostí danou limitem pro maximální rychlost na vstupu druhého bloku

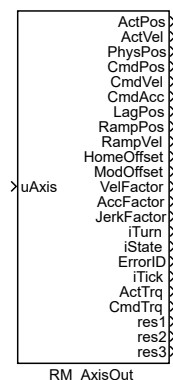
Ilustrace míchání pohybů



RM_AxisOut – Výstupní blok osy

Symbol bloku

Licence: [MOTION CONTROL](#)



Popis funkce

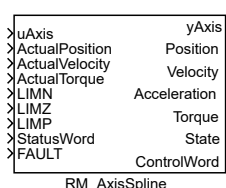
Blok **RM_AxisOut** zpřístupňuje důležité stavy bloku **RM_Axis**. Některé z výstupů jsou přímo na bloku **RM_Axis**, ale ty jsou o krok zpožděné. Aby nebyly výstupy zpožděné i na tomto bloku, je potřeba je zařadit jako poslední.

Poznámka: Téměř všechny bloky nepracují s momentem, proto je na příslušném výstupu 0. Obvykle je tento signál používán jako dopředná vazba pro regulátor rychlosti, tak to nepředstavuje problém.

RM_AxisSpline – Interpolace požadované polohy (rychlosti, zrychlení)

Symbol bloku

Licence: [MOTION CONTROL](#)



Popis funkce

Tento blok slouží k připojení virtuální osy (reprezentované blokem **RM_Axis**) k motoru respektive jeho řídicím obvodům a vytvoření fyzické osy. To zahrnuje několik do značné míry nezávislých funkcí, které realizuje tento blok.

Základní funkce je připojit požadované a aktuální signály o poloze, rychlosti, zrychlení a momentu. To pokrývají vstupy **ActualPosition**, **ActualVelocity**, **ActualTorque** a výstupy **Position**, **Velocity**, **Acceleration**, **Torque**.

Měníč motoru obvykle pracuje s jinými jednotkami (často přímo tiky na snímači polohy) než potřebujeme v ose. Dále polohu ještě ovlivňuje převodový poměr na převodovce. Blok **RM_AxisSpline** provádí transformaci jednotek podle parametrů **DriveUnits** a **AxisUnits** (zadáva se sobě odpovídající délka v jednotkách měniče a v jednotkách osy v bloku **RM_Axis**). Předpokládá se, že rychlost na vstupu i výstupu bloku **RM_AxisSpline** je drive units/s a zrychlení driveunits/s/s; moment se netransformuje.

Řídicí obvody motoru obvykle pracují s celočíselnými typy (respektive v celočíselných typech předávají hodnoty nadřazenému systému). Problém vzniká u polohy, kde při dlouhotrvajícím pohybu jedním směrem může dojít k přetečení celočíselného typu a hodnota se změní z maximálního použitelného čísla na minimální nebo naopak. Blok **RM_AxisSpline** s tímto umí pracovat a správně naváže polohu. Podmínkou správné funkce je, že v parametru **DriveBits** je nastaven správný počet bitů (včetně příznaku, jestli je číslo se znaménkem nebo bez) a také tuto vlastnost musí podporovat měnič (řídicí obvody motoru).

Měníče mají různé stavy a režimy činnosti a definované sekvence, jak se mezi těmito stavy přechází. Nejčastěji je to podle standardu CiA402. Blok **RM_AxisSpline** tento standard podporuje a to prostřednictvím vstupu **StatusWord**, výstupu **ControlWord** a parametrů **DriveMode**, **DriveTimeout** (vstup a výstup se připojuje na stejnojmenné signály v měniči, další je zřejmé z popisu parametrů).

Mnoho bloků pro řízení pohybu obsahuje výpočetně náročný algoritmus. To vede na relativně velké vzorkovací periody (typicky mezi 10 a 200 ms). Naproti tomu regulátor

motoru vyžaduje vzorkovací periodu malou (typicky do 1ms), aby nedocházelo k trhavému pohybu. Tyto protichůdné požadavky řeší blok `RM_AxisSpline`, který může běžet v jiné úloze (s kratší periodou vzorkování) než blok `RM_Axis` a provádí interpolaci hodnot, tak aby výsledná křivka byla spojitá a pokud možno hladká.

Je mnoho možností, jak dopočítat polohu (a rychlost, zrychlení, moment) v časech mezi hodnotami z pomalejší úlohy. Protože každá možnost má určité výhody a nevýhody, může si uživatel z několika vybrat pomocí parametru `InterpolationMode`. Podporované metody jsou:

- 1: **linear**: Poloha interpolována lineárně, rychlost jako derivace polohy, zrychlení 0 (tj. rychlost je po částech konstantní funkce se skoky).
- 2: **cubic spline**: Poloha je polynom 3. řádu, vypočtený na základě polohy a rychlosti na začátku a konci intervalu; rychlost je derivace polohy, zrychlení derivace rychlosti.
- 3: **quintic spline**: Poloha je polynom 5. řádu, vypočtený na základě polohy, rychlosti a zrychlení na začátku a konci intervalu; rychlost je derivace polohy, zrychlení derivace rychlosti.
- 4: **cubic approximation (B-spline)**: Poloha je polynom 3. řádu, vypočtený na základě dvou poloh před a dvou poloh za aktuálním intervalem; proložená funkce nemusí přesně procházet zadanými body; rychlost je derivace polohy, zrychlení derivace rychlosti.
- 5: **quintic approximation (B-spline)**: Poloha je polynom 5. řádu, vypočtený na základě tří poloh před a tří poloh za aktuálním intervalem; proložená funkce nemusí přesně procházet zadanými body; rychlost je derivace polohy, zrychlení derivace rychlosti.
- 6: **all linear**: Poloha, rychlost i zrychlení jsou interpolovány lineárně navzájem nezávisle, tj. rychlost neodpovídá přesně derivaci polohy a zrychlení neodpovídá přesně derivaci rychlosti.
- 7: **all cubic**: Poloha i rychlost jsou interpolovány polynomem 3. řádu navzájem nezávisle, tj. rychlost neodpovídá přesně derivaci polohy.
- 8: *rezervováno pro pozdější použití.*
- 9: *rezervováno pro pozdější použití.*

Nejjednodušší je lineární interpolace polohy. To však způsobuje skoky v rychlosti. Alternativně lze rychlost také interpolovat lineárně, ale pak zase rychlost neodpovídá derivaci/diferenci polohy. Z tohoto pohledu lepší je interpolace polynomem vyššího řádu (v parametru lze vybrat 3. nebo 5.), což ovšem vede na velké zrychlení, pokud vstupní trajektorie neodpovídá polynomu (nastává při přechodu z konstantního zrychlení na konstantní rychlost uprostřed mezi body vzorkování i při jednoduchých pohybech). Další

možnost je aproximace Bsplines funkcí (opět lze vybrat 3. nebo 5. řád). To odstraňuje nevýhody předchozích metod, ale vyžaduje více vzorků a tak zavádí větší dopravní zpoždění. Někdy také může vadit, že je to aproximační metoda a neprojíždí zadané body přesně (v režimu konstantní rychlosti je ale aproximace přesná). Ve všech případech je moment interpolován lineárně a nezávisle na ostatních veličinách.

Poznámka 1: Protože doba vykonávání bloků pro řízení pohybu značně kolísá, dostává interpolátor nové hodnoty neekvidistantně, přičemž výstup interpolátoru musí být souvislý. Proto se načte několik (závisí na metodě) hodnot dopředu a používají se hodnoty z vhodného intervalu. Právě používaný interval indikuje blok na výstupu **State** pomocí hodnot **interval1**, **interval2**, **interval3**. Důležité je pravidlo, že stavy odpovídající kladným číslům jsou normální činnost a stavy odpovídající záporným číslům představují chybu.

Poznámka 2: Protože činnost interpolátoru z principu vyžaduje použití v jiné (rychlejší) úloze než je **RM_Axis**, musí mít blok správně vyřešeno předávání reference mezi úlohami a referenci je možné bez obav předávat pomocí bloků **Inport** a **Outport**.

Vstup

uAxis	Odkaz na osu (přípustné je jen spojení RM_Axis.axisRef-uAxis nebo yAxis-uAxis)	Reference
ActualPosition	Skutečná poloha osy [drive unit]	Double (F64)
ActualVelocity	Skutečná rychlost osy [drive unit/s]	Double (F64)
ActualTorque	Skutečný moment/síla osy	Double (F64)
LIMN	Koncový spínač v záporném směru	Bool
LIMZ	Koncový spínač pro nulovou polohu	Bool
LIMP	Koncový spínač v kladném směru	Bool
StatusWord	Stavové slovo servoměniče dle CiA402 standardu	Long (I32)
FAULT	Vynucení vypnutí osy (obvykle chyba měniče nebo komunikace s měničem)	Bool

Výstupy

yAxis	Odkaz na osu (přípustné je jen spojení RM_Axis.axisRef-uAxis nebo yAxis-uAxis)	Reference
Position	Požadovaná (interpolovaná) poloha [drive unit]	Double (F64)
Velocity	Požadovaná (interpolovaná) rychlost [drive unit/s]	Double (F64)
Acceleration	Požadované (interpolované) zrychlení [drive unit/s/s]	Double (F64)
Torque	Požadovaný (interpolovaný) moment/síla	Double (F64)

State	Stav interpolátoru (záporné hodnoty znamenají chybu a pokud to není nesprávnou periodou, je potřeba zkontrolovat časování obou úloh)	Long (I32)
0	Neaktivní (interpolátor jen přeposílá vstupní data na výstup; nastává při ose ve stavu vypnuto nebo pokud bloky RM_Axis RM_AxisSpline mají stejnou periodu spouštění)	
1	Čekání (čekání, až bude dostatek dat v bufferu)	
2	Interval1 (probíhá interpolace, data z prvního intervalu)	
3	Interval2 (probíhá interpolace, data z prvního intervalu)	
x	Intervalx (další kladná čísla znamenají další interpolační intervaly; závisí na metodě, ale měla by se střídát jen 2 výjimečně 3 sousední čísla)	
-1	Přetečení (přetečení vyrovnávacího bufferu, tj. jsou odebírána pomaleji, než odpovídá nastavené periodě; algoritmus se restartuje automaticky, ale může dojít ke skoku polohy i rychlosti)	
-2	Podtečení (podtečení vyrovnávacího bufferu, tj. data přichází pomaleji, než odpovídá nastavené periodě; algoritmus se restartuje automaticky, ale může dojít ke skoku polohy i rychlosti)	
-3	Přetížení (data z bloku RM_Axis se nepodařilo přečíst; pravděpodobně je přetížena úloha s blokem RM_Axis)	
-4	Špatná perioda (úloha s blokem RM_AxisSpline má delší periodu než úloha s blokem RM_Axis)	
ControlWord	Řídící slovo servoměniče dle CiA402 standardu	Long (I32)

Parametry

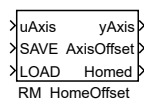
InterpolationMode	Algoritmus pro interpolaci	⊙9 Long (I32)
1	linear	
2	cubic spline	
3	quintic spline	
4	cubic aproximation (B-spline)	
5	quintic aproximation (B-spline)	
6	all linear	
7	all cubic	
8	—	
9	—	
ReverseLimit	Negace významu vstupů LIMN, LIMZ a LIMP	Bool

InterpolationMode	Režim řízení servoměniče	⊙9	Long (I32)
1	Zjednodušený CiA402 (ze statuswordu je kontrolován jen bit značící chybu, control word se nastavuje okamžitě bez postupné zapínací sekvence)		
2	Úplný CiA402 (úplná kontrola StatusWord v každém stavu, do ControlWord se posílá požadovaná sekvence a kontroluje se, jestli měnič na každý krok reaguje, pokud nereaguje do času daného parametrem DriveTimeout nastává chyba)		
DriveTimeout	Čas pro odpověď servoměniče [s] (for Strict CiA402 mode only)		Double (F64)
DriveBits	Počet platných bitů (záporné číslo znamená číslo se znaménkem) v polohovém vstupu ActualPosition a výstupu Position ↓-64 ↑63 ⊙-32		Long (I32)
DriveUnits	Vzdálenost v jednotkách měniče (pro transformaci polohy, vzdálenost odpovídající AxisUnits)		Double (F64)
AxisUnits	Vzdálenost v jednotkách osy (pro transformaci polohy, vzdálenost odpovídající DriveUnits)		Double (F64)
VelocityCalculate	když je zaškrtnuto, vstup ActualVelocity se neuplatňuje a rychlost je dopočítávána z rozdílu polohy (v aktuálním a předchozím kroku)		Bool

RM_HomeOffset – * Nastavení počáteční polohy pomocí offsetu

Symbol bloku

Licence: [MOTION CONTROL](#)



Popis funkce

Popis tohoto bloku ještě není k dispozici. Níže naleznete částečný popis vstupů, výstupů a parametrů bloku. Kompletní popis bloku bude k dispozici v dalších revizích dokumentace.

Tento blok nepropaguje kvalitu signálu. Více informací je uvedeno v sekci [1.4](#).

Vstup

uAxis	Odkaz na osu (přípustné je jen spojení RM_Axis.axisRef-uAxis nebo yAxis-uAxis)	Reference
SAVE	Nastavení offsetu z osy do bloku	Bool
LOAD	Nastavení offsetu z bloku do osy	Bool

Parametr

SavedOffset	Rozdíl mezi logickou a fyzickou polohou (pro nastavení/načtení z osy)	Double (F64)

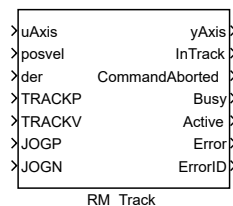
Výstup

yAxis	Odkaz na osu (přípustné je jen spojení RM_Axis.axisRef-uAxis nebo yAxis-uAxis)	Reference
AxisOffset	Aktuální rozdíl fyzické a logické polohy v ose	Bool
Homed	Příznak nastavení výchozí polohy	Bool

RM_Track – Sledování a krokování

Symbol bloku

Licence: [MOTION CONTROL](#)



Popis funkce

Blok **RM_Track** sdružuje několik užitečných doplňkových funkcí.

Pokud je aktivní (tj. nenulový) vstup **TRACK**, blok se snaží dosáhnout zadané polohy nebo rychlosti (vstup **posvel**) s respektováním omezení na rychlost, zrychlení/zpomalení a jerk. Rozdíl oproti bloku [MC_MoveAbsolute](#) je ten, že tento blok v každém kroku aktualizuje cílovou polohu nebo rychlost a přepočítává trajektorii. Tento režim je vhodný pro sledování trajektorie polohy nebo rychlosti generované mimo motion control subsystém. Pokud je trajektorie známa předem, je mnohem vhodnější a přesnější použít blok [MC_PositionProfile](#).

Pokud je aktivní (tj. nenulový) vstup **JOGP**, blok jede maximální dovolenou rychlostí v kladném směru s respektováním maximálního zrychlení a jerku při rozjezdu. Po deaktivaci signálu blok zastaví pohyb (s respektováním maximálního zpomalení a případně jerku) a vzdá se řízení osy. Tento režim je vhodný pro najetí na výchozí polohu operátorem pomocí tlačítek vpřed a vzad.

Vstup **JOGN** má stejnou funkci jako **JOGP**, jen je pohyb v negativním směru.

Pokud je na ose již nějaký pohyb aktivní, je tímto blokem přerušen (chová se tedy jako **BufferMode=aborting**). Pokud je sepnuto více funkcí, vykonává se ta s nejvyšší prioritou a ostatní jsou ignorovány. Pořadí je **TRACK**, **JOGP**, **JOGN**. Takovém režimu je však vhodné se vyhnout, protože chování není dostatečně intuitivní a výsledek je pak nečekaný.

Poznámka: pokud jsou všechny limity (rychlost, acc, dec, jerk) 0, limity se nekontrolují a jsou rovnou předány do osy. Vstupní signál však musí mít dostatečně malé zrychlení a rychlost, jinak osa přejde do stavu **ErrorStop**.

Vstupy

uAxis	Odkaz na osu (přípustné je jen spojení RM_Axis.axisRef-uAxis nebo yAxis-uAxis)	Reference
posvel	Požadovaná poloha nebo rychlost [unit, unit/s]	Double (F64)
TRACKP	Zapnutí režimu sledování polohy	Bool
TRACKV	Zapnutí režimu sledování rychlosti	Bool

JOGP	Zapnutí režimu posunu v kladném směru	Bool
JOGN	Zapnutí režimu posuvu v záporném směru	Bool

Výstupy

yAxis	Odkaz na osu (přípustné je jen spojení RM_Axis.axisRef-uAxis nebo yAxis-uAxis)	Reference
InTrack	Příznak dosažení požadované polohy v režimu sledování polohy	Bool
CommandAborted	Příznak přerušení funkce bloku	Bool
Busy	Příznak, že algoritmus ještě neskončil	Bool
Active	Příznak, že blok řídí osu	Bool
Error	Příznak chyby	Bool
ErrorID	Výsledek poslední operace i obecná chyba systému REXYGEN	Error

Parametry

pv	Maximální povolená rychlost [unit/s]	Double (F64)
pa	Maximální povolené zrychlení [unit/s ²]	Double (F64)
pd	Maximální povolené zpomalení [unit/s ²]	Double (F64)
pj	Maximální povolená změna zrychlení [unit/s ³]	Double (F64)
iLen	Počet kroků pro odhad rychlosti	⊙10 Long (I32)

Kapitola 21

MC_MULTI – Řízení pohybu více os

Obsah

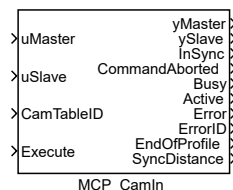
MCP_CamIn – * Zapnutí vačky	763
MCP_CamTableSelect – Definice vačky	765
MCP_CombineAxes – * Kombinace pohybu dvou os do třetí	767
MCP_GearIn – * Zapnutí konstantního převodového poměru	769
MCP_GearInPos – * Zapnutí konstantního převodového poměru v zadané pozici	771
MCP_PhasingAbsolute – * Vytvoření fázového posunu (absolutní souřadnice)	773
MCP_PhasingRelative – * Vytvoření fázového posunu (relativně k pozici při spuštění)	775
MC_CamIn, MCP_CamIn – Zapnutí vačky	777
MC_CamOut – Vypnutí vačky	781
MC_CombineAxes, MCP_CombineAxes – Kombinace pohybu dvou os do třetí	783
MC_GearIn, MCP_GearIn – Zapnutí konstantního převodového poměru	786
MC_GearInPos, MCP_GearInPos – Zapnutí konstantního převodového poměru v zadané pozici	789
MC_GearOut – Vypnutí konstantního převodového poměru	794
MC_PhasingAbsolute, MCP_PhasingAbsolute – Vytvoření fázového posunu (absolutní souřadnice)	796
MC_PhasingRelative, MCP_PhasingRelative – Vytvoření fázového posunu (relativně k pozici při spuštění)	799

Knihovna MC_MULTI se specializuje na řízení pohybu ve více osách. Obsahuje bloky jako [MC_CombineAxes](#) pro synchronizaci více os, [MC_GearIn](#) a [MC_GearOut](#) pro

převody a `MC_PhasingAbsolute`, `MC_PhasingRelative` pro přesné fázování os. Knihovna nabízí `MC_CamIn` a `MC_CamOut` pro funkce vačky, které umožňují sledování složitých profilů pohybu. `MCP_CamTableSelect` navíc poskytuje flexibilitu při výběru vačkových tabulek a `MC_GearInPos` umožňuje řazení na základě polohy. Tato knihovna je nezbytná pro pokročilé aplikace vyžadující koordinované řízení pohybu napříč více osami.

MCP_CamIn — * **Zapnutí vačky**

Symbol bloku

Licence: [MOTION CONTROL](#)

Popis funkce

Popis tohoto bloku ještě není k dispozici. Níže naleznete částečný popis vstupů, výstupů a parametrů bloku. Kompletní popis bloku bude k dispozici v dalších revizích dokumentace.

Tento blok nepropaguje kvalitu signálu. Více informací je uvedeno v sekci [1.4](#).

Vstup

uMaster	Odkaz na hlavní osu	Reference
uSlave	Odkaz na podřízenou osu	Reference
CamTableID	Odkaz na vačku (spojit s MCP_CamTableSelect.CamTableID)	Reference
Execute	Náběžná hrana aktivuje blok	Bool

Parametr

MasterOffset	Posunutí v definici vačky na straně hlavní osy [unit]	Double (F64)
SlaveOffset	Posunutí v definici vačky na straně podřízené osy [unit]	Double (F64)
MasterScaling	Měřítko pro definici vačky (strana hlavní osy)	⊙1.0 Double (F64)
SlaveScaling	Měřítko pro definici vačky (strana podřízené osy)	⊙1.0 Double (F64)
StartMode	Volba absolutního nebo relativního profilu vačky	⊙4 Long (I32)
	1 hlavní osa relativní podřízená osa absolutní	
	2 hlavní osa absolutní podřízená osa relativní	
	3 obě osy relativní	
	4 obě osy absolutní	
BufferMode	Režim převzetí osy	⊙1 Long (I32)
	1 aborting	
	2 buffered	
	3 blending low	
	4 blending high	
	5 blending previous	
	6 blending next	
RampIn	RampIn faktor (0 = RampIn režim se nepoužívá)	Double (F64)

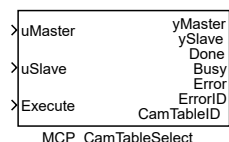
Výstup

yMaster	Odkaz na hlavní osu	Reference
ySlave	Odkaz na podřízenou osu	Reference
InSync	Příznak dosažení profilu vačky podřízenou osou	Bool
CommandAborted	Příznak přerušení funkce bloku	Bool
Busy	Příznak, že algoritmus ještě neskončil	Bool
Active	Příznak, že blok řídí osu	Bool
Error	Příznak chyby	Bool
ErrorID	Výsledek poslední operace i kód chyby REXYGENu	Error
EndOfProfile	Příznak dosažení konce profilu vačky (jen necyklické vačky)	Bool
SyncDistance	Odchylka v poloze podřízené osy od synchronizované polohy	Double (F64)

MCP_CamTableSelect – Definice vačky

Symbol bloku

Licence: [MOTION CONTROL](#)



Popis funkce

Blok `MCP_CamTableSelect` spolupracuje s blokem `MCP_CamIn` a definuje vačku jako spojitou funkci jedné proměnné. Možnosti definování této funkce jsou analogické, jako v bloku `MC_PositionProfile`, tj. máme dvě možnosti:

1. tabulkou: zadávají se dvojice čísel poloha hlavní osy a poloha podřízené osy. Mezi jednotlivými časy se poloha interpoluje polynomem třetího řádu (lineární interpolaci lze také zvolit, ale není příliš vhodná, protože na okrajích intervalu pak je skok v rychlosti). Hodnoty polohy hlavní osy se zadávají do pole/parametru `mValues`, příslušné hodnoty polohy podřízené osy do pole/parametru `sValues`. Posloupnost hodnot `mValues` musí být stoupající.

2. polynomy: celá funkce se v hlavní ose (tj. v nezávislé proměnné) rozdělí na několik intervalů a pro každý interval se zadá aproximující polynom pátého řádu. Polohy hlavní osy a tím i příslušné intervaly se definují jako v předchozím případě v poli `mValues`. Polynom pro každý interval je ve tvaru $p(x) = a_5x^5 + a_4x^4 + a_3x^3 + a_2x^2 + a_1x + a_0$, přičemž na začátku intervalu je $x = 0$, a na konci $x = 1$. Koeficienty a_i jsou uloženy v poli `sValues` ve vzestupném pořadí (tj. pole `sValues` obsahuje 6 hodnot pro každý časový interval). Tato metoda umožňuje snížit počet intervalů a pro určení koeficientů polynomů existuje speciální grafický editor.

Pro obě varianty je možné zvolit rozdělení na stejně dlouhé intervaly, pak je v poli `mValues` jen počáteční a koncová poloha.

Poznámka: pokud je parametr `alg=1` nebo `alg=5` nebo `alg=9` je možné nechat pole `mValues` prázdné (popř. nepřipojený vstup) a pak zadávat celou vačku v poli `sValues`, kde první sloupec jsou polohy master osy, druhý sloupec polohy slave osy, třetí (nepovinný) sloupec derivace, čtvrtý (nepovinný) sloupec druhá derivace.

Vstupy

<code>uMaster</code>	Odkaz na hlavní osu	Reference
<code>uSlave</code>	Odkaz na podřízenou osu	Reference
<code>Execute</code>	Náběžná hrana aktivuje blok	Bool

Výstupy

yMaster	Odkaz na hlavní osu	Reference
ySlave	Odkaz na podřízenou osu	Reference
Done	Příznak dokončení algoritmu	Bool
Busy	Příznak, že algoritmus ještě neskončil	Bool
Error	Příznak chyby	Bool
ErrorID	Výsledek poslední operace	Error
CamTableID	Odkaz na vačku (spojit s MC_CamIn.CamTableID)	Reference
	i obecná chyba systému REXYGEN	

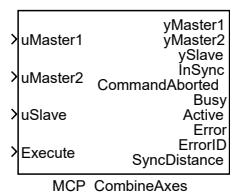
Parametry

camname	Jméno souboru, kam si speciální editor ukládá data (pokud je parametr prázdný, zvolí se automaticky podle jména bloku)	String
nmax	Maximální počet segmentů profilu	⊙3 Long (I32)
alg	Typ interpolace	⊙2 Long (I32)
	1 tabulka čas/hodnota (interpolace polynomem 3. řádu)	
	2 hodnoty ve stejném intervalu (interpolace polynomem 3. řádu)	
	3 interpolace polynomy 5. řádu	
	4 polynomy 5. řádu s ekvidistantními intervaly	
	5 tabulka čas/hodnota + počáteční a koncová derivace	
	6 hodnoty ve stejném intervalu + počáteční a koncová derivace	
	7 hodnoty ve stejném intervalu (aproximace B-splinem 3. řádu)	
	8 hodnoty ve stejném intervalu (aproximace B-splinem 5. řádu)	
	9 tabulka čas/hodnota (lineární interpolace)	
Periodic	Příznak cyklické vačky (konec navazuje na začátek)	⊙on Bool
BeginRate	Počáteční strmost (derivace ds/dm) vačky (jen pro alg=5 a alg=6)	Double (F64)
EndRate	Počáteční strmost (derivace ds/dm) vačky (jen pro alg=5 a alg=6)	Double (F64)
mValues	Posloupnost hraničních pozic jednotlivých segmentů na hlavní ose	Double (F64) ⊙[0 30]
sValues	Posloupnost poloh řízené osy nebo koeficienty interpolačních polynomů (a0, a1, a2, ...)	Double (F64) ⊙[0 100 100 0]

MCP_CombineAxes — * Kombinace pohybu dvou os do třetí

Symbol bloku

Licence: [MOTION CONTROL](#)



Popis funkce

Popis tohoto bloku ještě není k dispozici. Níže naleznete částečný popis vstupů, výstupů a parametrů bloku. Kompletní popis bloku bude k dispozici v dalších revizích dokumentace.

Tento blok nepropaguje kvalitu signálu. Více informací je uvedeno v sekci [1.4](#).

Vstup

uMaster1	Odkaz na první hlavní osu	Reference
uMaster2	Odkaz na druhou hlavní osu	Reference
uSlave	Odkaz na podřízenou osu	Reference
Execute	Náběžná hrana aktivuje blok	Bool

Parametr

GearRatioNumeratorM1	Čítec převodového poměru master osy 1	⊙1	Long (I32)
GearRatioDenominatorM1	Jmenovatel převodového poměru master osy 1		Long (I32)
		⊙1	
GearRatioNumeratorM2	Čítec převodového poměru master osy 2	⊙1	Long (I32)
GearRatioDenominatorM2	Jmenovatel převodového poměru master osy 2		Long (I32)
		⊙1	
BufferMode	Režim převzetí osy	⊙1	Long (I32)
	1 aborting		
	2 buffered		
	3 blending low		
	4 blending high		
	5 blending previous		
	6 blending next		
CombineMode	režim sloučení os	⊙1	Long (I32)
	1 součet		
	2 rozdíl		
RampIn	RampIn faktor (0 = RampIn režim se nepoužívá)		Double (F64)

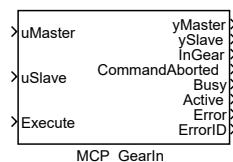
Výstup

yMaster1	Odkaz na první hlavní osu	Reference
yMaster2	Odkaz na druhou hlavní osu	Reference
ySlave	Odkaz na podřízenou osu	Reference
InSync	Příznak dosažení profilu vačky podřízenou osou	Bool
CommandAborted	Příznak přerušení funkce bloku	Bool
Busy	Příznak, že algoritmus ještě neskončil	Bool
Active	Příznak, že blok řídí osu	Bool
Error	Příznak chyby	Bool
ErrorID	Výsledek poslední operace i kód chyby REXYGENu	Error
SyncDistance	Odchylka v poloze podřízené osy od synchronizované polohy	Double (F64)

MCP_GearIn — * Zapnutí konstantního převodového poměru

Symbol bloku

Licence: [MOTION CONTROL](#)



Popis funkce

Popis tohoto bloku ještě není k dispozici. Níže naleznete částečný popis vstupů, výstupů a parametrů bloku. Kompletní popis bloku bude k dispozici v dalších revizích dokumentace.

Tento blok nepropaguje kvalitu signálu. Více informací je uvedeno v sekci [1.4](#).

Vstup

uMaster	Odkaz na hlavní osu	Reference
uSlave	Odkaz na podřízenou osu	Reference
Execute	Náběžná hrana aktivuje blok	Bool

Parametr

RatioNumerator	Převodový poměr - čítec (podřízená osa)	⊙1	Long (I32)
RatioDenominator	Převodový poměr - jmenovatel (hlavní osa)	⊙1	Long (I32)
Acceleration	Maximální povolené zrychlení [unit/s ²]		Double (F64)
Deceleration	Maximální povolené zpomalení [unit/s ²]		Double (F64)
Jerk	Maximální povolená změna zrychlení [unit/s ³]		Double (F64)
BufferMode	Režim převzetí osy	⊙1	Long (I32)
	1 aborting		
	2 buffered		
	3 blending low		
	4 blending high		
	5 blending previous		
	6 blending next		

Výstup

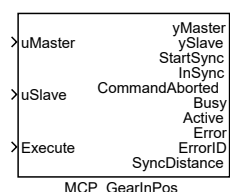
yMaster	Odkaz na hlavní osu	Reference
ySlave	Odkaz na podřízenou osu	Reference
InGear	Příznak dosažení požadované rychlosti řízenou osou	Bool
CommandAborted	Příznak přerušení funkce bloku	Bool
Busy	Příznak, že algoritmus ještě neskončil	Bool

Active	Příznak, že blok řídí osu	Bool
Error	Příznak chyby	Bool
ErrorID	Výsledek poslední operace i kód chyby REXYGENu	Error

MCP_GearInPos – * Zapnutí konstantního převodového poměru v zadané pozici

Symbol bloku

Licence: [MOTION CONTROL](#)



Popis funkce

Popis tohoto bloku ještě není k dispozici. Níže naleznete částečný popis vstupů, výstupů a parametrů bloku. Kompletní popis bloku bude k dispozici v dalších revizích dokumentace.

Tento blok nepropaguje kvalitu signálu. Více informací je uvedeno v sekci [1.4](#).

Vstup

uMaster	Odkaz na hlavní osu	Reference
uSlave	Odkaz na podřízenou osu	Reference
Execute	Náběžná hrana aktivuje blok	Bool

Parametr

RatioNumerator	Převodový poměr - čítec (podřízená osa)	⊙1	Long (I32)
RatioDenominator	Převodový poměr - jmenovatel (hlavní osa)	⊙1	Long (I32)
MasterSyncPosition	Poloha hlavní osy v okamžiku zasynchronizování [unit]		Double (F64)
SlaveSyncPosition	Poloha podřízené osy v okamžiku zasynchronizování		Double (F64)
MasterStartDistance	Definuje polohu hlavní osy pro spuštění procesu synchronizace		Double (F64)
Velocity	Maximální povolená rychlost [unit/s]		Double (F64)
Acceleration	Maximální povolené zrychlení [unit/s^2]		Double (F64)
Deceleration	Maximální povolené zpomalení [unit/s^2]		Double (F64)
Jerk	Maximální povolená změna zrychlení [unit/s^3]		Double (F64)
BufferMode	Režim převzetí osy	⊙1	Long (I32)
	1 aborting		
	2 buffered		
	3 blending low		
	4 blending high		
	5 blending previous		
	6 blending next		

SyncMode	Režim synchronizace (pouze cyklické osy)	⊙2	Long (I32)
	1 synchronizovat zrychlením		
	2 synchronizovat na nejbližší bod		
	3 synchronizovat zpomalením		

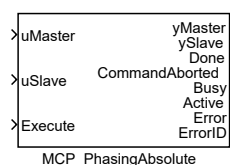
Výstup

yMaster	Odkaz na hlavní osu	Reference
ySlave	Odkaz na podřízenou osu	Reference
StartSync	Začíná synchronizace pohybu	Bool
InSync	Příznak dosažení profilu vačky podřízenou osou	Bool
CommandAborted	Příznak přerušení funkce bloku	Bool
Busy	Příznak, že algoritmus ještě neskončil	Bool
Active	Příznak, že blok řídí osu	Bool
Error	Příznak chyby	Bool
ErrorID	Výsledek poslední operace	Error
	i kód chyby REXYGENu	
SyncDistance	Odchylka v poloze podřízené osy od synchronizované polohy	Double (F64)

MCP_PhasingAbsolute – * Vytvoření fázového posunu (absolutní souřadnice)

Symbol bloku

Licence: [MOTION CONTROL](#)



Popis funkce

Popis tohoto bloku ještě není k dispozici. Níže naleznete částečný popis vstupů, výstupů a parametrů bloku. Kompletní popis bloku bude k dispozici v dalších revizích dokumentace.

Tento blok nepropaguje kvalitu signálu. Více informací je uvedeno v sekci [1.4](#).

Vstup

uMaster	Odkaz na hlavní osu	Reference
uSlave	Odkaz na podřízenou osu	Reference
Execute	Náběžná hrana aktivuje blok	Bool

Parametr

PhaseShift	Požadovaný fázový posuv (vzdálenost na hlavní ose) vačky	Double (F64)
Velocity	Maximální povolená rychlost [unit/s]	Double (F64)
Acceleration	Maximální povolené zrychlení [unit/s ²]	Double (F64)
Deceleration	Maximální povolené zpomalení [unit/s ²]	Double (F64)
Jerk	Maximální povolená změna zrychlení [unit/s ³]	Double (F64)
BufferMode	Režim převzetí osy	⊙1 Long (I32)
	1 aborting	
	2 buffered	

Výstup

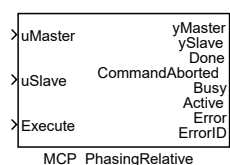
yMaster	Odkaz na hlavní osu	Reference
ySlave	Odkaz na podřízenou osu	Reference
Done	Příznak dokončení algoritmu	Bool
CommandAborted	Příznak přerušení funkce bloku	Bool
Busy	Příznak, že algoritmus ještě neskončil	Bool
Active	Příznak, že blok řídí osu	Bool
Error	Příznak chyby	Bool

ErrorID	Výsledek poslední operace	Error
i	 kód chyby REXYGENu	

MCP_PhasingRelative – * Vytvoření fázového posunu (relativně k pozici při spuštění)

Symbol bloku

Licence: [MOTION CONTROL](#)



Popis funkce

Popis tohoto bloku ještě není k dispozici. Níže naleznete částečný popis vstupů, výstupů a parametrů bloku. Kompletní popis bloku bude k dispozici v dalších revizích dokumentace.

Tento blok nepropaguje kvalitu signálu. Více informací je uvedeno v sekci [1.4](#).

Vstup

uMaster	Odkaz na hlavní osu	Reference
uSlave	Odkaz na podřízenou osu	Reference
Execute	Náběžná hrana aktivuje blok	Bool

Parametr

PhaseShift	Požadovaný fázový posuv (vzdálenost na hlavní ose) vačky	Double (F64)
Velocity	Maximální povolená rychlost [unit/s]	Double (F64)
Acceleration	Maximální povolené zrychlení [unit/s ²]	Double (F64)
Deceleration	Maximální povolené zpomalení [unit/s ²]	Double (F64)
Jerk	Maximální povolená změna zrychlení [unit/s ³]	Double (F64)
BufferMode	Režim převzetí osy	⊙1 Long (I32)
	1 aborting	
	2 buffered	

Výstup

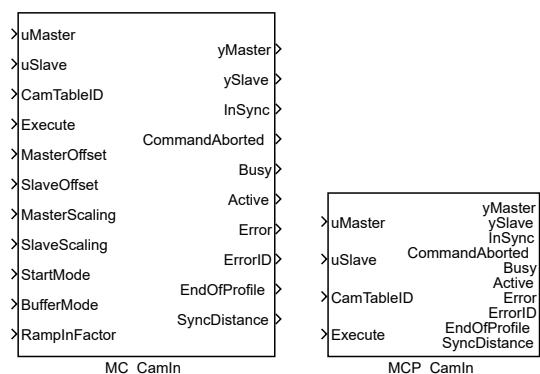
yMaster	Odkaz na hlavní osu	Reference
ySlave	Odkaz na podřízenou osu	Reference
Done	Příznak dokončení algoritmu	Bool
CommandAborted	Příznak přerušení funkce bloku	Bool
Busy	Příznak, že algoritmus ještě neskončil	Bool
Active	Příznak, že blok řídí osu	Bool
Error	Příznak chyby	Bool

ErrorID	Výsledek poslední operace	Error
i	 kód chyby REXYGENu	

MC_CamIn, MCP_CamIn – Zapnutí vačky

Symbyly bloků

Licence: [MOTION CONTROL](#)



Popis funkce

Bloky MC_CamIn a MCP_CamIn mají naprosto shodnou funkci, jediným rozdílem je, že MCP_ varianta bloku má méně vstupů a potřebné konstanty se zadávají jako parametry bloku.

Blok MC_CamIn zapíná režim, kdy je podřízená osa (tj. ta, která je připojena ke vstupu uSlave) řízena tak, že její poloha je závislá na poloze hlavní osy (tj. ta, která je připojena ke vstupu uMaster), přičemž převodní funkce je určena blokem [MCP_CamTableSelect](#) připojeným ke vstupu CamTableID. Pokud převodní funkci označíme $Cam(\cdot)$, polohu hlavní osy $PosM$ a polohu podřízené osy $PosS$, pak platí (pro absolutní režim, bez fázování):

$$PosS = Cam((PosM - MasterOffset) / MasterScaling) * SlaveScaling + SlaveOffset$$

Tento režim osy je často nazýván elektronická vačka.

Režim vačky lze ukončit zapnutím jiného pohybu na podřízené ose v režimu **aborting** nebo spuštěním bloku [MC_CamOut](#). Pokud vačka (její definiční funkce - viz popis bloku není cyklická [MCP_CamTableSelect](#)), dojde k ukončení režimu vačky také při vyjetí hlavní osy z definičního oboru funkce vačky. Toto je signalizováno výstupem **EndOfProfile**.

Při aktivaci funkce vačky (tj. v okamžiku, kdy blok [MC_CamIn](#) převeze řízení osy) nemusí poloha a rychlost (popř. i zrychlení, ale současná implementace při zapnutí vačky jerk neuvažuje a připouští skok ve zrychlení) odpovídat požadovaným hodnotám, tj. poloze a rychlosti hlavní osy a profilu vačky. V takovém případě záleží na hodnotě parametru **RampIn**. Pokud má parametr hodnotu 0, režim vačky se nezapne a je signalizována

chyba -707 (skok v poloze nebo rychlosti). Pokud je parametr kladný, nastává přechodový děj, kdy poloha ještě neodpovídá poloze podle definice vačky - tzv. RampIn režim. Parametr **RampIn** přibližně odpovídá rychlosti, kterou by se podřízená osa pohybovala během RampIn režimu, kdyby hlavní osa stála. Pokud se hlavní osa pohybuje, pak je výsledný pohyb určen součtem pohybu daného definicí vačky a pohybu daného RampIn režimem se stojící hlavní osou. Pokud je potřeba RampIn režim použít, volíme parametr **RampIn** přibližně 0,1 až 0,5 maximální rychlosti podřízené osy a pokud dojde k chybě překročení maximální rychlosti nebo zrychlení během RampIn režimu, tak jej zmenšíme.

Vstupy

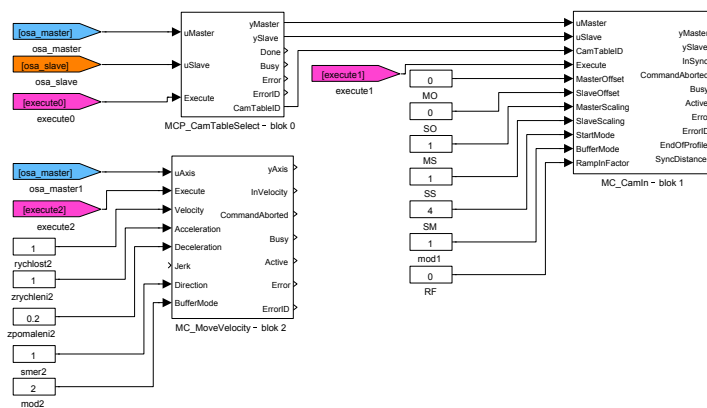
uMaster	Odkaz na hlavní osu	Reference
uSlave	Odkaz na podřízenou osu	Reference
CamTableID	Odkaz na vačku (spojit s MCP_CamTableSelect.CamTableID)	Reference
Execute	Náběžná hrana aktivuje blok	Bool
MasterOffset	Posunutí v definici vačky na straně hlavní osy [unit]	Double (F64)
SlaveOffset	Posunutí v definici vačky na straně podřízené osy [unit]	Double (F64)
MasterScaling	Měřítko pro definici vačky (strana hlavní osy)	Double (F64)
SlaveScaling	Měřítko pro definici vačky (strana podřízené osy)	Double (F64)
StartMode	Volba absolutního nebo relativního profilu vačky	Long (I32)
	1 hlavní osa relativní podřízená osa absolutní	
	2 hlavní osa absolutní podřízená osa relativní	
	3 obě osy relativní	
	4 obě osy absolutní	
BufferMode	Režim převzetí osy	Long (I32)
	1 Aborting (nový blok se spustí okamžitě)	
	2 Buffered (nový blok se spustí po dokončení předchozího)	
	3 Blending low (nový blok se spustí po dokončení předchozího, původní pohyb skončí s nižší rychlostí z obou bloků)	
	4 Blending high (nový blok se spustí po dokončení předchozího, původní pohyb skončí s vyšší rychlostí z obou bloků)	
	5 Blending previous (nový blok se spustí po dokončení předchozího, původní pohyb skončí se svojí koncovou rychlostí)	
	6 Blending next (nový blok se spustí po dokončení předchozího, původní pohyb skončí s počáteční rychlostí nového bloku)	
RampIn	RampIn faktor (0 = RampIn režim se nepoužívá); odpovídá přibližně rychlosti synchronizace [unit/s] podřízené osy na polohu vačky	Double (F64)

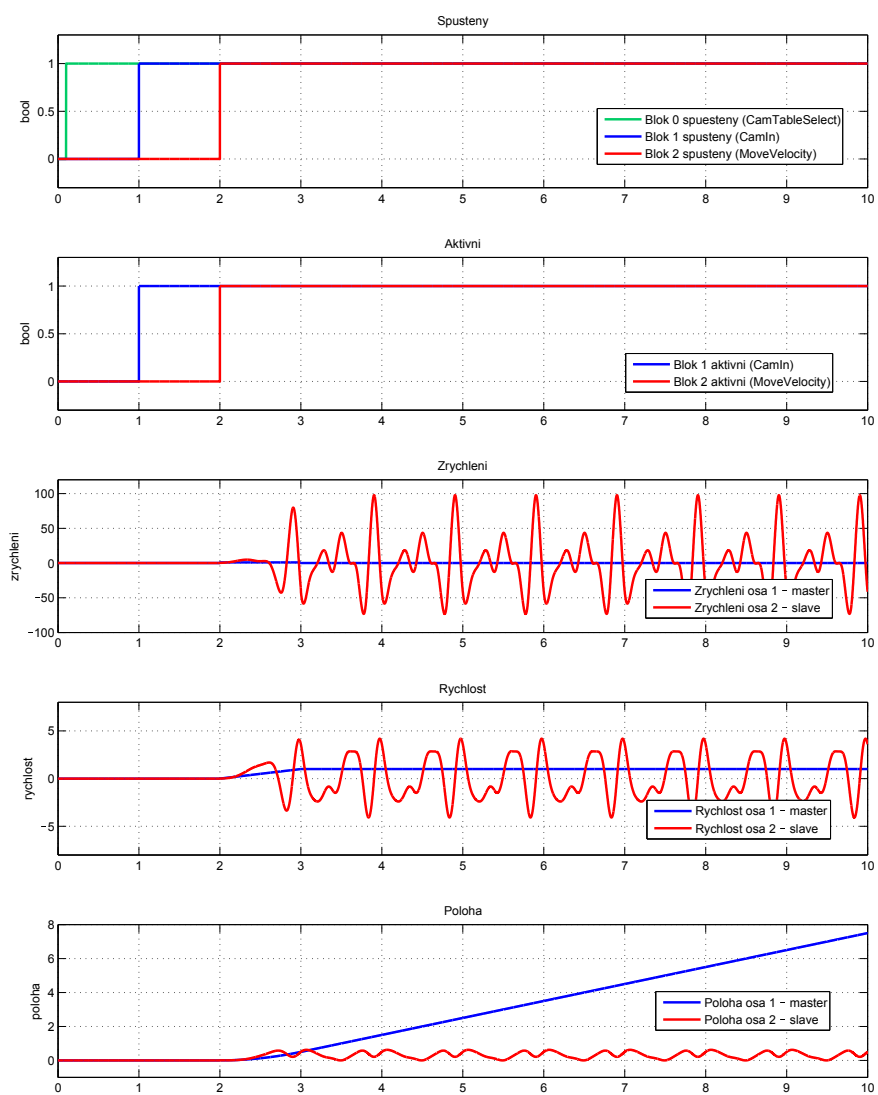
Výstupy

yMaster	Odkaz na hlavní osu	Reference

ySlave	Odkaz na podřízenou osu	Reference
InSync	Příznak dosažení profilu vačky podřízenou osou	Bool
CommandAborted	Příznak přerušení funkce bloku	Bool
Busy	Příznak, že algoritmus ještě neskončil	Bool
Active	Příznak, že blok řídí osu	Bool
Error	Příznak chyby	Bool
ErrorID	Výsledek poslední operace i obecná chyba systému REXYGEN	Error
EndOfProfile	Příznak dosažení konce profilu vačky (u necyklické vačky konec pohybu)	Bool
SyncDistance	Odchylka v poloze podřízené osy od synchronizované polohy	Double (F64)

Příklady

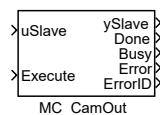




MC_CamOut – Vypnutí vačky

Symbol bloku

Licence: [MOTION CONTROL](#)



Popis funkce

Blok **MC_CamOut** ukončuje režim vačky zapnutý blokem [MC_CamIn](#). Pokud žádná vačka není aktivní, blok nemá žádnou funkci (a ani nehlásí chybu).

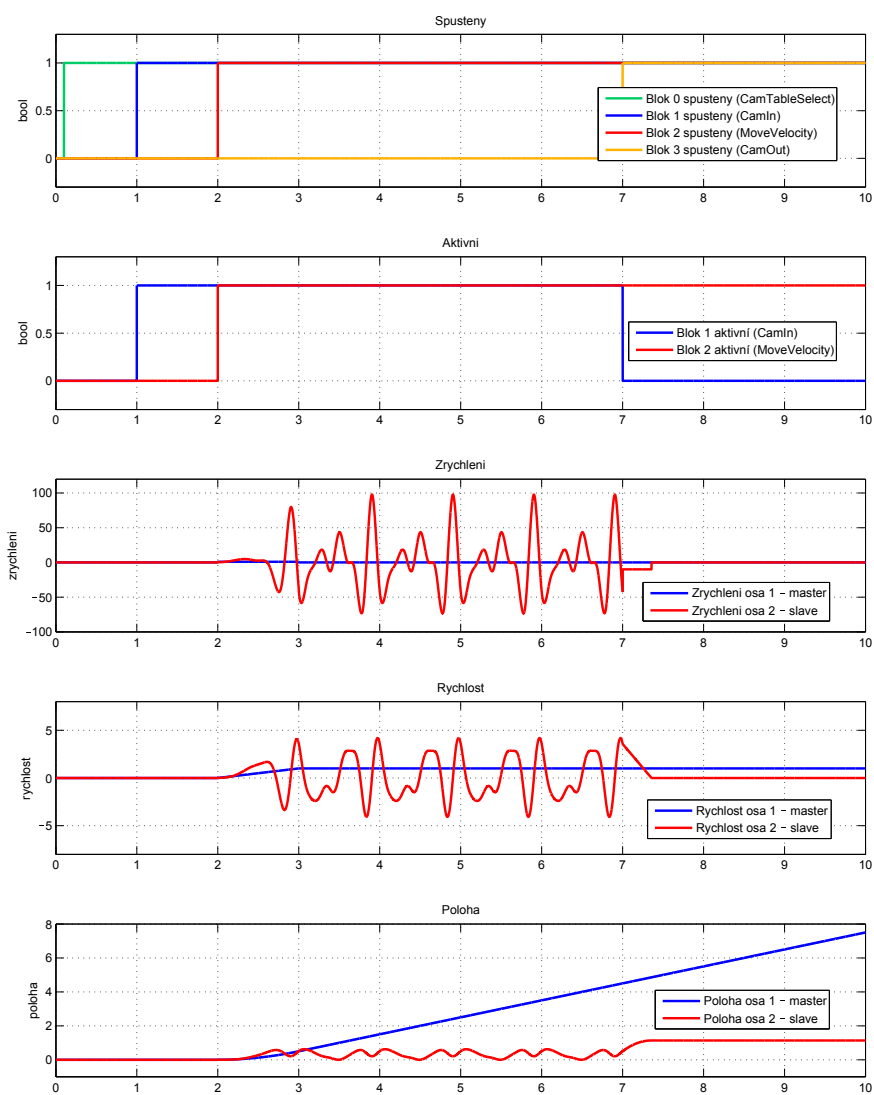
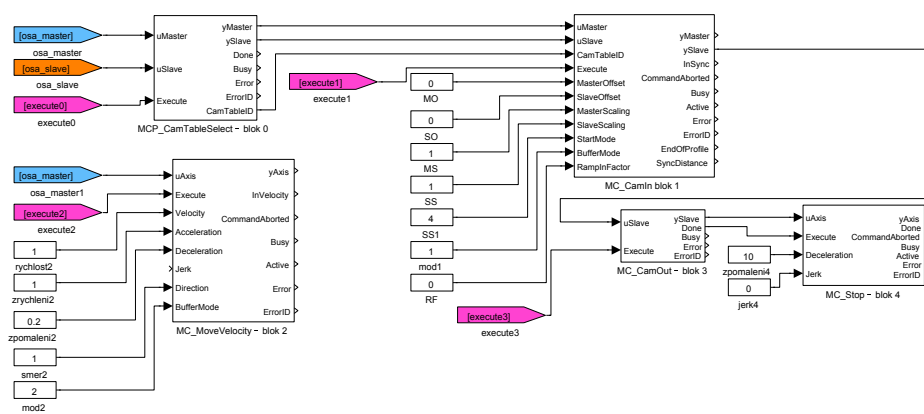
Vstupy

uSlave	Odkaz na podřízenou osu	Reference
Execute	Náběžná hrana aktivuje blok	Bool

Výstupy

ySlave	Odkaz na podřízenou osu	Reference
Done	Příznak dokončení algoritmu	Bool
Busy	Příznak, že algoritmus ještě neskončil	Bool
Error	Příznak chyby	Bool
ErrorID	Výsledek poslední operace i obecná chyba systému REXYGEN	Error

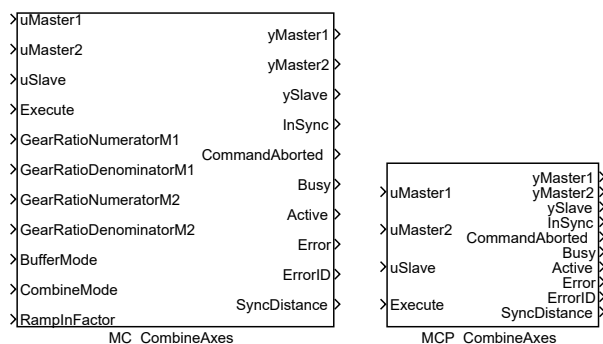
Příklad



MC_CombineAxes, MCP_CombineAxes – Kombinace pohybu dvou os do třetí

Symboly bloků

Licence: [MOTION CONTROL](#)



Popis funkce

Bloky MC_CombineAxes a MCP_CombineAxes mají naprosto shodnou funkci, jediným rozdílem je, že MCP_ varianta bloku má méně vstupů a potřebné konstanty se zadávají jako parametry bloku.

Blok MC_CombineAxes kombinuje pohyb dvou os do třetí. V podstatě se jedná o výpočet nové žádané pozice na základě dvou poloh. Platí, že

$$\begin{aligned} \text{SlavePosition} = & \text{Master1Position} \cdot \frac{\text{GearRatioNumeratorM1}}{\text{GearRatioDenominatorM1}} + \\ & + \text{Master2Position} \cdot \frac{\text{GearRatioNumeratorM2}}{\text{GearRatioDenominatorM2}} \end{aligned}$$

Blok umožňuje zadat do parametru GearRatio... záporné číslo a výsledný pohyb podřízené osy může být rozdíl poloh obou hlavních os. Primárně by se měl rozdíl realizovat pomocí parametru CombineMode

Vstupy

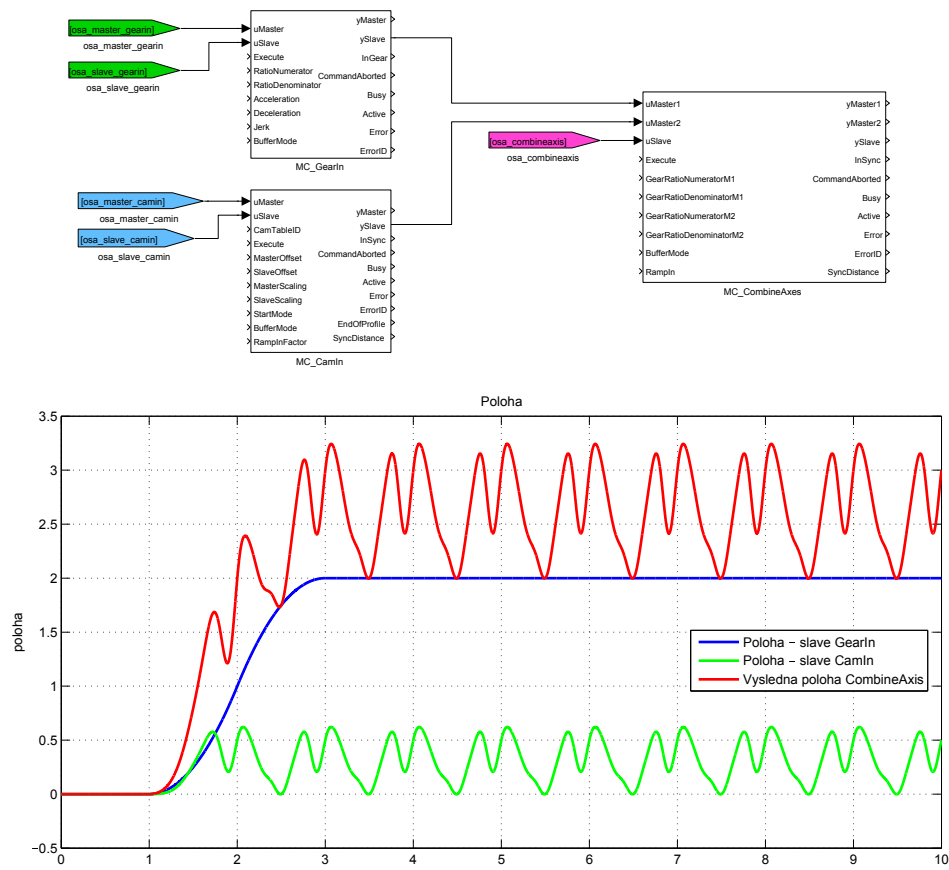
uMaster1	Odkaz na první hlavní osu	Reference
uMaster2	Odkaz na druhou hlavní osu	Reference
uSlave	Odkaz na podřízenou osu	Reference
Execute	Náběžná hrana aktivuje blok	Bool
GearRatioNumeratorM1	Číselník převodového poměru master osy 1	Long (I32)
GearRatioDenominatorM1	Jmenovatel převodového poměru master osy 1	Long (I32)
GearRatioNumeratorM2	Číselník převodového poměru master osy 2	Long (I32)

GearRatioDenominatorM2	Jmenovatel převodového poměru master osy 2	Long (I32)
BufferMode	Režim převzetí osy	Long (I32)
	1 Aborting (nový blok se spustí okamžitě)	
	2 Buffered (nový blok se spustí po dokončení předchozího)	
	3 Blending low (nový blok se spustí po dokončení předchozího, původní pohyb skončí s nižší rychlostí z obou bloků)	
	4 Blending high (nový blok se spustí po dokončení předchozího, původní pohyb skončí s vyšší rychlostí z obou bloků)	
	5 Blending previous (nový blok se spustí po dokončení předchozího, původní pohyb skončí se svojí koncovou rychlostí)	
	6 Blending next (nový blok se spustí po dokončení předchozího, původní pohyb skončí s počáteční rychlostí nového bloku)	
CombineMode	Režim sloučení os	⊙1 Long (I32)
	1 součet	
	2 rozdíl	
RampInFactor	RampIn faktor (0 = RampIn režim se nepoužívá); rychlost synchronizace [unit/s] podřízené osy na polohu vačky odpovídá přibližně $\text{RampIn} * \text{MaxVelocityAppl}$, kde MaxVelocityAppl se zadává v bloku RM_Axis připojeného ke vstupu uSlave; analogicky zrychlení	Double (F64)

Výstupy

yMaster1	Odkaz na první hlavní osu	Reference
yMaster2	Odkaz na druhou hlavní osu	Reference
ySlave	Odkaz na podřízenou osu	Reference
InSync	Příznak dosažení profilu vačky podřízenou osou	Bool
CommandAborted	Příznak přerušení funkce bloku	Bool
Busy	Příznak, že algoritmus ještě neskončil	Bool
Active	Příznak, že blok řídí osu	Bool
Error	Příznak chyby	Bool
ErrorID	Výsledek poslední operace	Error
	i obecná chyba systému REXYGEN	
SyncDistance	Odchylka v poloze podřízené osy od synchronizované polohy	Double (F64)

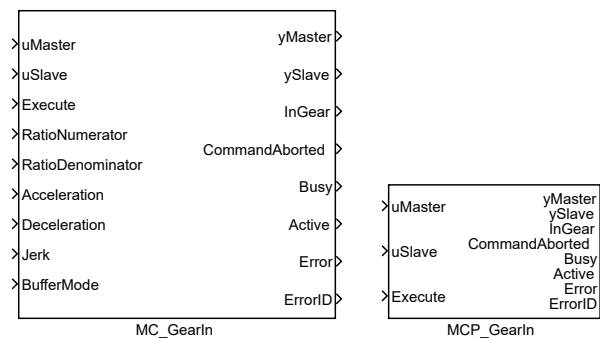
Příklad



MC_GearIn, MCP_GearIn – Zapnutí konstantního převodového poměru

Symboly bloků

Licence: [MOTION CONTROL](#)



Popis funkce

Bloky MC_GearIn a MCP_GearIn mají naprosto shodnou funkci, jediným rozdílem je, že MCP_ varianta bloku má méně vstupů a potřebné konstanty se zadávají jako parametry bloku.

Blok MC_GearIn zapíná režim, kdy je podřízená osa (tj. ta, která je připojena ke vstupu uSlave) řízena tak, že její poloha je závislá na poloze hlavní osy (tj. ta, která je připojena ke vstupu uMaster), přičemž poměr rychlostí obou os je dán parametry RatioNumerator a RatioDenominator. Pokud označíme rychlost hlavní osy $VelM$ a rychlost podřízené osy $VelS$, pak platí (bez fázování):

$$VelS = VelM \cdot \frac{RatioNumerator}{RatioDenominator}$$

Tento režim osy je často nazýván elektronická převodovka. Poloha a zrychlení podřízené osy je dopočítávána konzistentně s uvedenou rychlostí.

Režim převodovky lze ukončit zapnutím jiného pohybu na podřízené ose v režimu **aborting** nebo spuštěním bloku [MC_GearOut](#).

Při aktivaci funkce převodovky (tj. v okamžiku, kdy blok [MC_GearIn](#) převeze řízení osy) nemusí rychlost (a popřípadě i zrychlení pokud jej požadujeme spojitě, tj. $jerk <> 0$) odpovídat požadované hodnotě rychlosti (a popř. i zrychlení) hlavní osy a převodovému poměru. V takovém případě nastává přechodový děj, kdy rychlost ještě neodpovídá převodovému poměru - tzv. RampIn režim. V tomto RampIn režimu jsou použity parametry Acceleration, Deceleration, Jerk a blok řídí podřízenou osu tak, aby se co nejdříve (s respektováním omezení na zrychlení a popř. jerk) dostala do synchronního stavu.

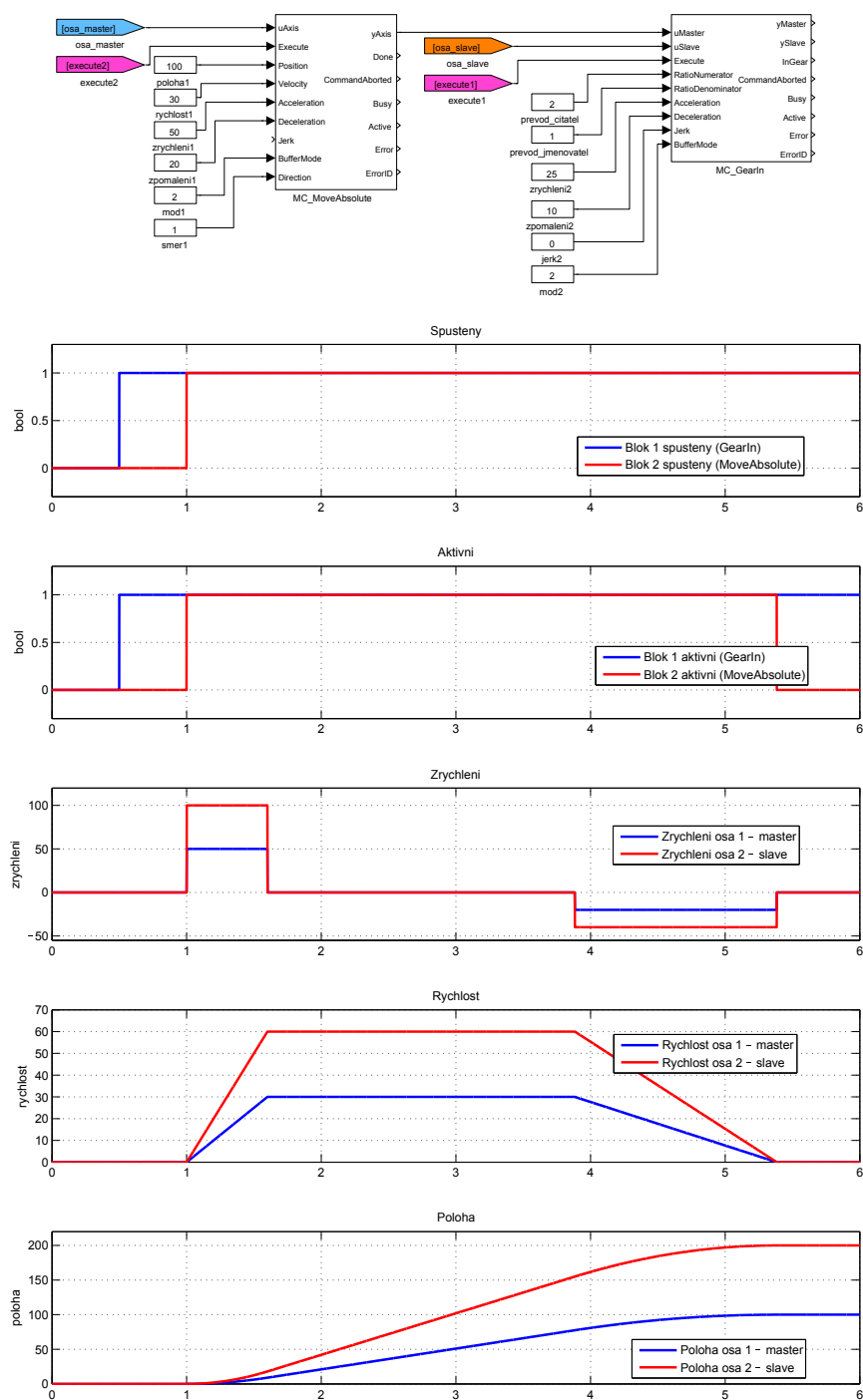
Vstupy

uMaster	Odkaz na hlavní osu	Reference
uSlave	Odkaz na podřízenou osu	Reference
Execute	Náběžná hrana aktivuje blok	Bool
RatioNumerator	Převodový poměr - čítec (podřízená osa)	Long (I32)
RatioDenominator	Převodový poměr - jmenovatel (hlavní osa)	Long (I32)
Acceleration	Maximální povolené zrychlení [unit/s ²]	Double (F64)
Deceleration	Maximální povolené zpomalení [unit/s ²]	Double (F64)
Jerk	Maximální povolená změna zrychlení [unit/s ³]	Double (F64)
BufferMode	Režim převzetí osy	Long (I32)
	1 Aborting (nový blok se spustí okamžitě)	
	2 Buffered (nový blok se spustí po dokončení předchozího)	
	3 Blending low (nový blok se spustí po dokončení předchozího, původní pohyb skončí s nižší rychlostí z obou bloků)	
	4 Blending high (nový blok se spustí po dokončení předchozího, původní pohyb skončí s vyšší rychlostí z obou bloků)	
	5 Blending previous (nový blok se spustí po dokončení předchozího, původní pohyb skončí se svojí koncovou rychlostí)	
	6 Blending next (nový blok se spustí po dokončení předchozího, původní pohyb skončí s počáteční rychlostí nového bloku)	

Výstupy

yMaster	Odkaz na hlavní osu	Reference
ySlave	Odkaz na podřízenou osu	Reference
InGear	Příznak dosažení požadované rychlosti řízenou osou	Bool
CommandAborted	Příznak přerušení funkce bloku	Bool
Busy	Příznak, že algoritmus ještě neskončil	Bool
Active	Příznak, že blok řídí osu	Bool
Error	Příznak chyby	Bool
ErrorID	Výsledek poslední operace	Error
	i obecná chyba systému REXYGEN	

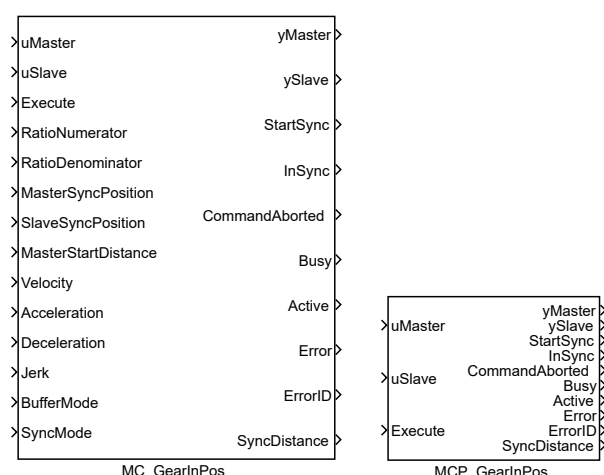
Příklad



MC_GearInPos, MCP_GearInPos – Zapnutí konstantního převodového poměru v zadané pozici

Symbyoly bloků

Licence: [MOTION CONTROL](#)



Popis funkce

Bloky MC_GearInPos a MCP_GearInPos mají naprosto shodnou funkci, jediným rozdílem je, že MCP_ varianta bloku má méně vstupů a potřebné konstanty se zadávají jako parametry bloku.

Blok **MC_GearInPos** zapne řízení podřízené osy tak, že poměr rychlostí hlavní a podřízené osy je v poměru `RatioNumerator:RatioDenominator`. Na rozdíl od bloku **MC_GearIn** je u tohoto bloku zachováván i poměr vzdáleností a je určena i poloha obou os v okamžiku zasynchronizování, tzn. platí

$$\frac{SlavePosition - SlaveSyncPosition}{MasterPosition - MasterSyncPosition} = \frac{RatioNumerator}{RatioDenominator}$$

Pokud v okamžiku aktivace bloku (tj. v okamžiku náběžné hrany na vstupu `Execute` a v případě buffered režimu ještě po ukončení předchozích bloků) neodpovídá poloha podřízené osy poloze hlavní osy (dle výše uvedeného vzorce), je spuštěn proces synchronizace (indikováno výstupem `StartSync`). Během tohoto procesu je generována trajektorie tak, aby došlo co nejrychleji k synchronizaci a nebyly překročeny maximální hodnoty rychlostí, zrychlení a změny zrychlení dané parametry `Velocity`, `Acceleration`, `Deceleration`, `Jerk`. Po dokončení synchronizace se již tyto limity neuplatňují. Pokud je parameter `MasterStartDistance=0`, proces synchronizace se spouští hned po aktivaci bloku (vstu-

pem Execute). V opačném případě se proces synchronizace spouští ve chvíli, kdy hlavní osa dosáhne polohy v intervalu $\text{MasterSyncPosition} \pm \text{MasterStartDistance}$.

Poznámky:

1. Proces synchronizace používá dva algoritmy: I. Algoritmus shodný s blokem **MC_MoveAbsolute**, přičemž trajektorie je v každém kroku přepočítána tak, aby koncová rychlost odpovídala aktuální rychlosti hlavní osy. II. Poloha (i rychlost i zrychlení) se generuje jako v případě synchronního pohybu, ale k hodnotám je po určitou dobu přičítána hodnota vhodného polynomu 5. stupně, tak aby nenastal skok v poloze, v rychlosti a v případě zadaného jerku ani ve zrychlení a aby na konci doby přičítání měl polynom nulovou hodnotu. Doba je volena tak, aby polynom nepřekročil požadované meze rychlosti, zrychlení a jerku. První metoda nevede k úspěšné synchronizaci, pokud se hlavní osa pohybuje se zrychlením, druhá metoda negarantuje dodržení limitů na rychlost, zrychlení a jerk, navíc vyžaduje ponechat určitou rychlost a zrychlení na synchronní část, takže je obecně delší. Proto se oba algoritmy vhodně kombinují.

2. Parametry bloku (ať už start procesu synchronizace nebo limity na rychlost a zrychlení) je nutné volit tak, aby podřízená osa byla v poloze **SlaveSyncPosition** (přibližně) ve stejný okamžik, jako hlavní osa v poloze **MasterSyncPosition**. V opačné případě může nastat neočekávaný pohyb podřízené osy nebo porušení zadaných limitů. Pohyb hlavní osy může být libovolný, ale v konkrétní aplikaci bývá obvykle dobře definován. Správným nastavením parametrů je tedy možné zajistit vhodný průběh synchronizace.

Vstupy

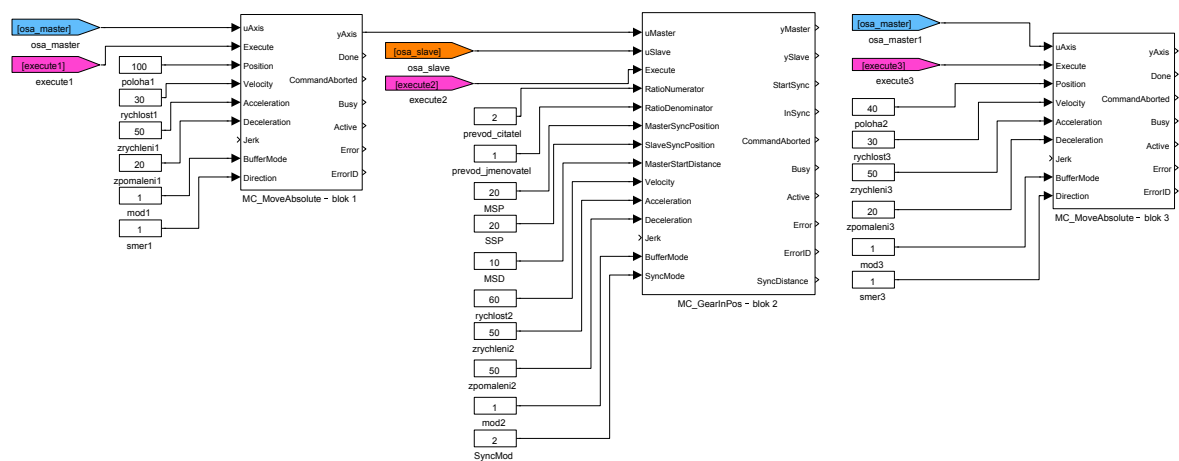
uMaster	Odkaz na hlavní osu	Reference
uSlave	Odkaz na podřízenou osu	Reference
Execute	Náběžná hrana aktivuje blok	Bool
RatioNumerator	Převodový poměr - čítec (podřízená osa)	Long (I32)
RatioDenominator	Převodový poměr - jmenovatel (hlavní osa)	Long (I32)
MasterSyncPosition	Poloha hlavní osy v okamžiku zasynchronizování [unit]	Double (F64)
SlaveSyncPosition	Poloha podřízené osy v okamžiku zasynchronizování	Double (F64)
MasterStartDistance	Definuje polohu hlavní osy pro spuštění procesu synchronizace	Double (F64)
Velocity	Maximální povolená rychlost [unit/s]	Double (F64)
Acceleration	Maximální povolené zrychlení [unit/s ²]	Double (F64)
Deceleration	Maximální povolené zpomalení [unit/s ²]	Double (F64)
Jerk	Maximální povolená změna zrychlení [unit/s ³]	Double (F64)

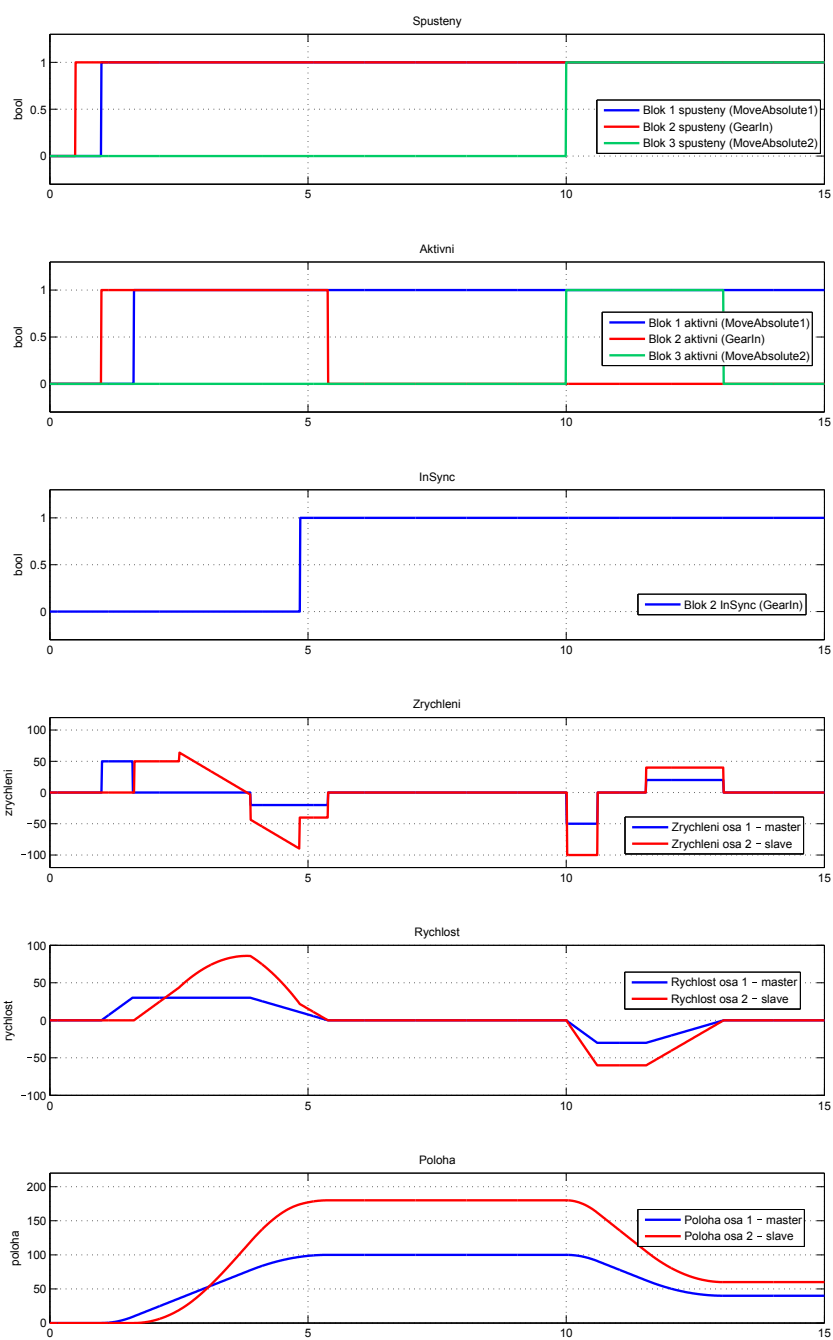
BufferMode	Režim převzetí osy	Long (I32)
1	 Aborting (nový blok se spustí okamžitě)	
2	 Buffered (nový blok se spustí po dokončení předchozího)	
3	 Blending low (nový blok se spustí po dokončení předchozího, původní pohyb skončí s nižší rychlostí z obou bloků)	
4	 Blending high (nový blok se spustí po dokončení předchozího, původní pohyb skončí s vyšší rychlostí z obou bloků)	
5	 Blending previous (nový blok se spustí po dokončení předchozího, původní pohyb skončí se svojí koncovou rychlostí)	
6	 Blending next (nový blok se spustí po dokončení předchozího, původní pohyb skončí s počáteční rychlostí nového bloku)	
SyncMode	Režim synchronizace (pouze cyklické osy)	Long (I32)
1	 synchronizovat zrychlením	
2	 synchronizovat na nejbližší bod	
3	 synchronizovat zpomalením	

Výstupy

yMaster	Odkaz na hlavní osu	Reference
ySlave	Odkaz na podřízenou osu	Reference
StartSync	Začíná synchronizace pohybu	Bool
InSync	Příznak dosažení profilu vačky podřízenou osou	Bool
CommandAborted	Příznak přerušení funkce bloku	Bool
Busy	Příznak, že algoritmus ještě neskončil	Bool
Active	Příznak, že blok řídí osu	Bool
Error	Příznak chyby	Bool
ErrorID	Výsledek poslední operace	Error
	i obecná chyba systému REXYGEN	
SyncDistance	Odchylka v poloze podřízené osy od synchronizované polohy [unit/s]	Double (F64)

Příklad

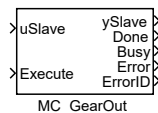




MC_GearOut – Vypnutí konstantního převodového poměru

Symbol bloku

Licence: [MOTION CONTROL](#)



Popis funkce

Blok **MC_GearOut** ukončuje režim elektronické převodovky zapnutý blokem [MC_GearIn](#) nebo [MC_GearInPos](#) nebo [MC_CombineAxis](#). Pokud režim elektronické převodovky není aktivní, blok hlásí chybu -703 (Invalid state).

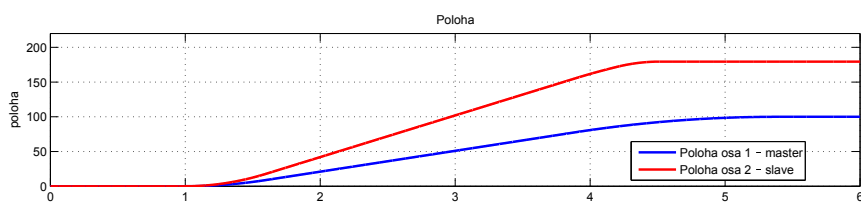
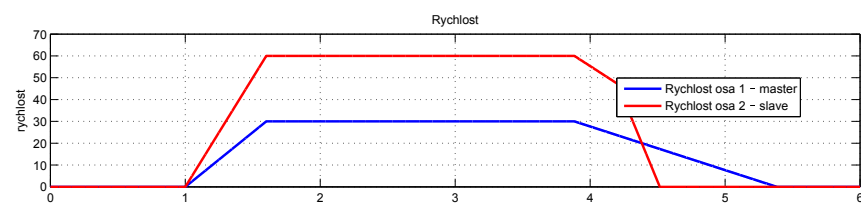
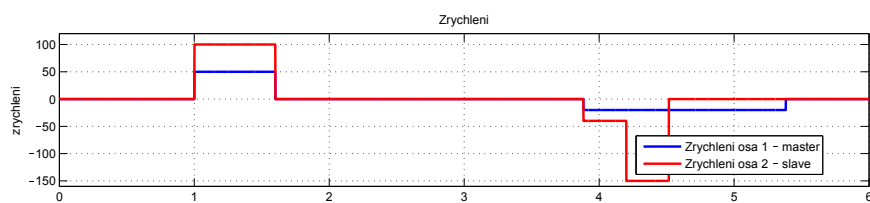
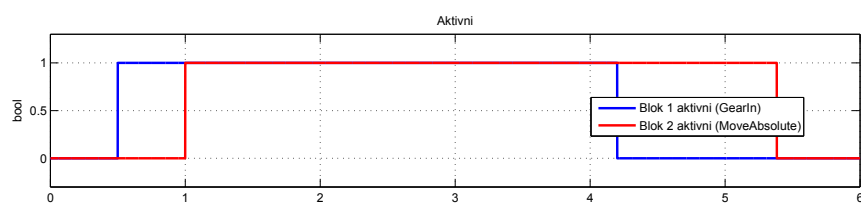
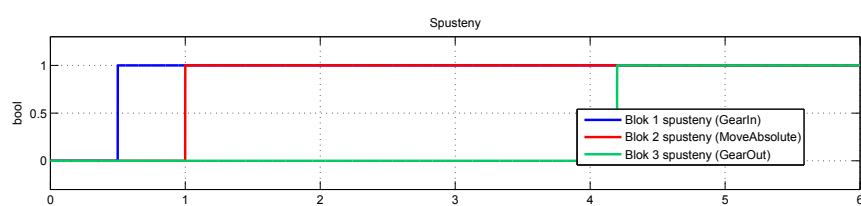
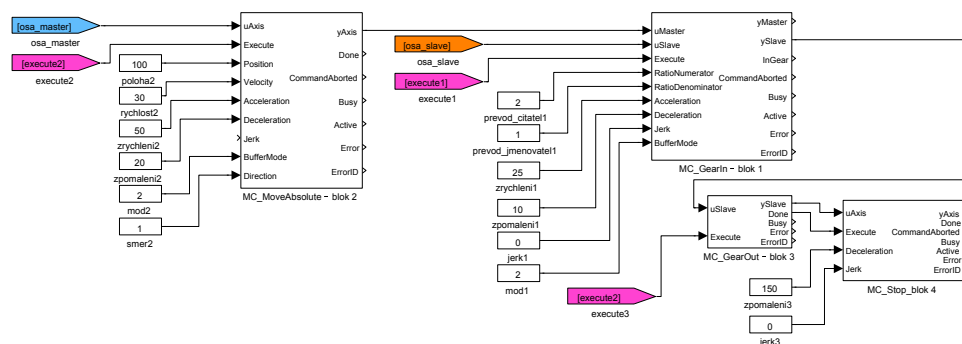
Vstupy

uSlave	Odkaz na podřízenou osu	Reference
Execute	Náběžná hrana aktivuje blok	Bool

Výstupy

ySlave	Odkaz na podřízenou osu	Reference
Done	Příznak dokončení algoritmu	Bool
Busy	Příznak, že algoritmus ještě neskončil	Bool
Error	Příznak chyby	Bool
ErrorID	Výsledek poslední operace	Error
	i obecná chyba systému REXYGEN	

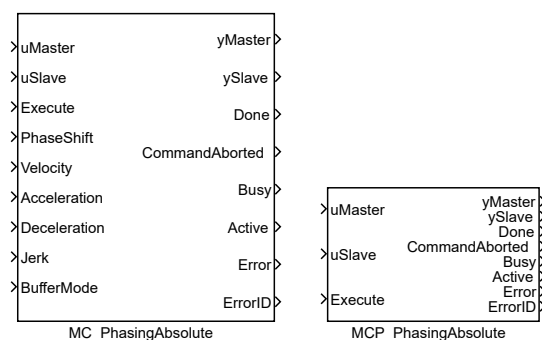
Příklad



MC_PhasingAbsolute, MCP_PhasingAbsolute – Vytvoření fázového posunu (absolutní souřadnice)

Symbody bloků

Licence: [MOTION CONTROL](#)



Popis funkce

Bloky MC_PhasingAbsolute a MCP_PhasingAbsolute mají naprosto shodnou funkci, jediným rozdílem je, že MCP_ varianta bloku má méně vstupů a potřebné konstanty se zadávají jako parametry bloku.

Blok **MC_PhasingAbsolute** zavádí další posunutí na hlavní ose pro vačku (blok **MC_CamIn**) a převodovku (blok **MC_GearIn**). Blok funguje velice podobně bloku **MC_MoveSuperimposed** (tj. generuje pohyb z bodu 0 do bodu **PhaseShift** s respektováním omezení na rychlost, zrychlení a popřípadě jerk, tak aby pohyb trval co nejkratší dobu) s tím rozdílem, že generovaná poloha/rychlost/zrychlení se nepřičítá ke skutečné poloze hlavní osy, ale přičítá se k ní jen z pohledu bloku **MC_CamIn**, **MC_GearIn**, **MC_GearInPos**, **MC_CombineAxis**.

Poznámka 1: Tento blok je analogie natočení mechanické vačky na hřídeli o úhel **PhaseShift**.

Poznámka 2: Pokud již na ose nějaké fázové posunutí je, tak hodnota **PhaseShift** je výsledné posunutí. Fázové posunutí se nuluje, pokud (slave) osa přejde do stavu disabled nebo je spuštěn další pohyb v režimu aborting.

Vstupy

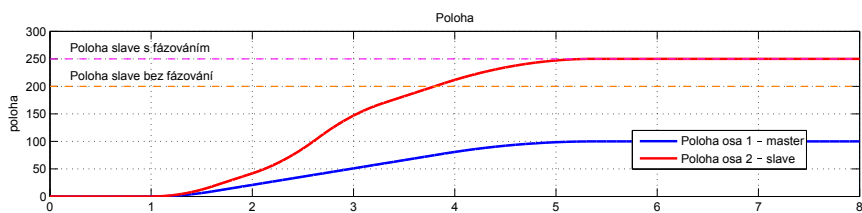
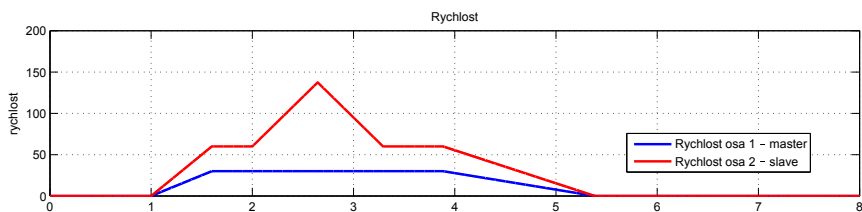
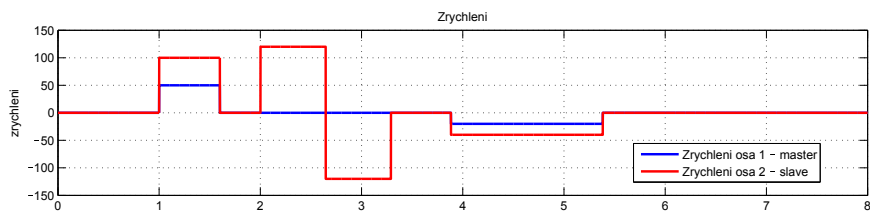
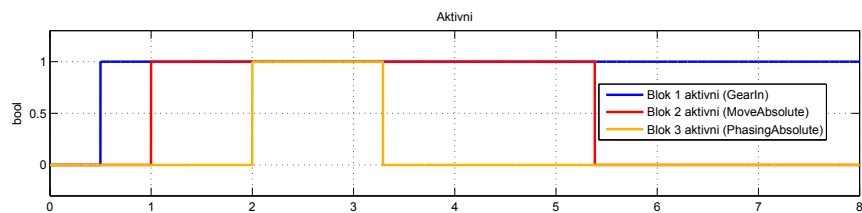
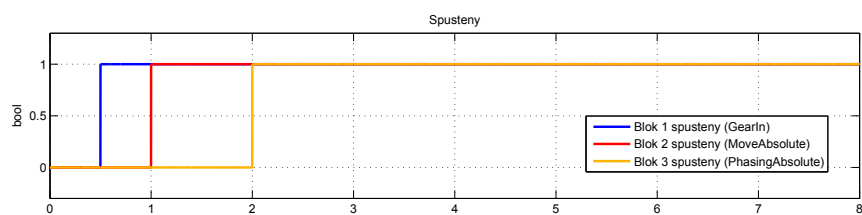
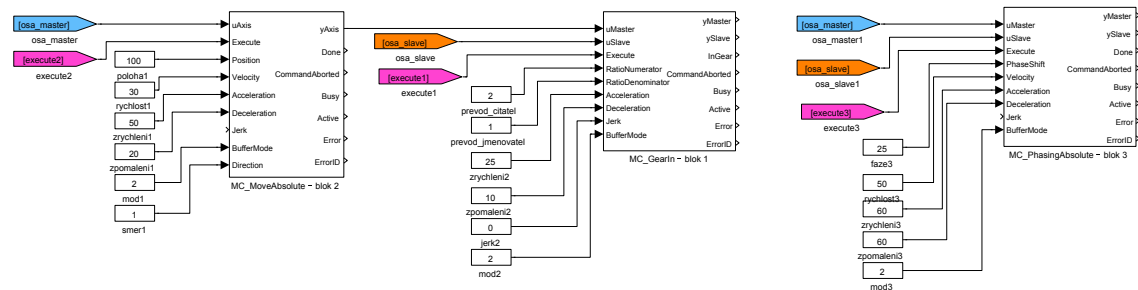
uMaster	Odkaz na hlavní osu	Reference
uSlave	Odkaz na podřízenou osu	Reference
Execute	Náběžná hrana aktivuje blok	Bool
PhaseShift	Požadovaný fázový posuv (vzdálenost na hlavní ose) vačky	Double (F64)
Velocity	Maximální povolená rychlost [unit/s]	Double (F64)
Acceleration	Maximální povolené zrychlení [unit/s ²]	Double (F64)

Deceleration	Maximální povolené zpomalení [unit/s ²]	Double (F64)
Jerk	Maximální povolená změna zrychlení [unit/s ³]	Double (F64)
BufferMode	Režim převzetí osy	Long (I32)
	1 aborting	
	2 buffered	

Výstupy

yMaster	Odkaz na hlavní osu	Reference
ySlave	Odkaz na podřízenou osu	Reference
Done	Příznak dokončení algoritmu	Bool
CommandAborted	Příznak přerušení funkce bloku	Bool
Busy	Příznak, že algoritmus ještě neskončil	Bool
Active	Příznak, že blok řídí osu	Bool
Error	Příznak chyby	Bool
ErrorID	Výsledek poslední operace	Error
	i obecná chyba systému REXYGEN	

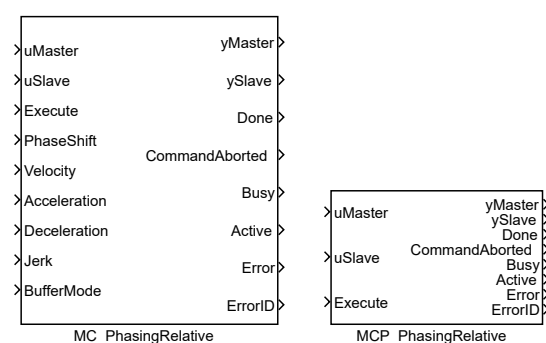
Příklad



MC_PhasingRelative, MCP_PhasingRelative – Vytvoření fázového posunu (relativně k pozici při spuštění)

Symbody bloků

Licence: [MOTION CONTROL](#)



Popis funkce

Bloky MC_PhasingRelative a MCP_PhasingRelative mají naprosto shodnou funkci, jediným rozdílem je, že MCP_ varianta bloku má méně vstupů a potřebné konstanty se zadávají jako parametry bloku.

Blok **MC_PhasingRelative** zavádí další posunutí na hlavní ose pro vačku (blok **MC_CamIn**) a převodovku (blok **MC_GearIn**). Koncová poloha se určí tak, že se k aktuální poloze v okamžiku spuštění (tj. náběžné hrany na vstupu **Execute**) přičte hodnota parametru **PhaseShift**. Blok funguje velice podobně bloku **MC_MoveSuperimposed** (tj. generuje pohyb z bodu 0 do bodu **PhaseShift** s respektováním omezení na rychlost, zrychlení a popřípadě jerk, tak aby pohyb trval co nejkratší dobu) s tím rozdílem, že generovaná poloha/rychlost/zrychlení se nepřičítá ke skutečné poloze hlavní osy, ale přičítá se k ní jen z pohledu bloku **MC_CamIn**, **MC_GearIn**, **MC_GearInPos**, **MC_CombineAxis**.

Poznámka 1: Tento blok je analogie natočení mechanické vačky na hřídeli o úhel **PhaseShift**.

Poznámka 2: Pokud již na ose nějaké fázové posunutí je, tak se hodnota **PhaseShift** přičítá k existujícímu posunutí. Fázové posunutí se nuluje, pokud (slave) osa přejde do stavu disabled nebo je spuštěn další pohyb v režimu aborting.

Vstupy

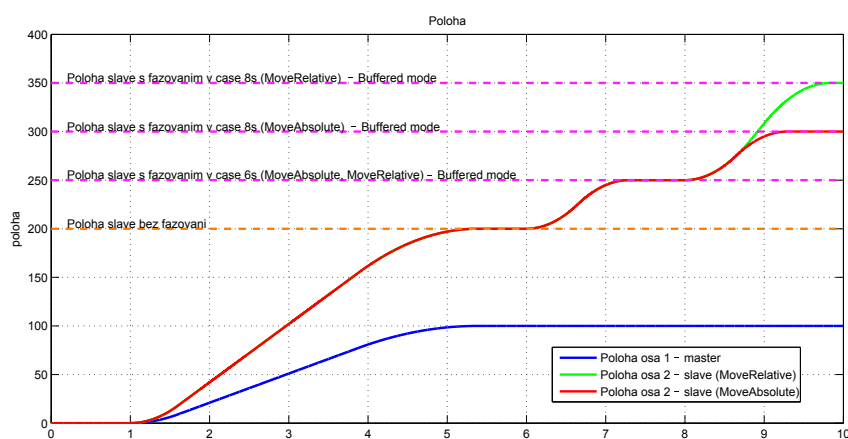
uMaster	Odkaz na hlavní osu	Reference
uSlave	Odkaz na podřízenou osu	Reference
Execute	Náběžná hrana aktivuje blok	Bool
PhaseShift	Požadovaný fázový posuv (vzdálenost na hlavní ose) vačky	Double (F64)

Velocity	Maximální povolená rychlost [unit/s]	Double (F64)
Acceleration	Maximální povolené zrychlení [unit/s ²]	Double (F64)
Deceleration	Maximální povolené zpomalení [unit/s ²]	Double (F64)
Jerk	Maximální povolená změna zrychlení [unit/s ³]	Double (F64)
BufferMode	Režim převzetí osy	Long (I32)
	1 aborting	
	2 buffered	

Výstupy

yMaster	Odkaz na hlavní osu	Reference
ySlave	Odkaz na podřízenou osu	Reference
Done	Příznak dokončení algoritmu	Bool
CommandAborted	Příznak přerušení funkce bloku	Bool
Busy	Příznak, že algoritmus ještě neskončil	Bool
Active	Příznak, že blok řídí osu	Bool
Error	Příznak chyby	Bool
ErrorID	Výsledek poslední operace	Error
	i obecná chyba systému REXYGEN	

Příklad



Kapitola 22

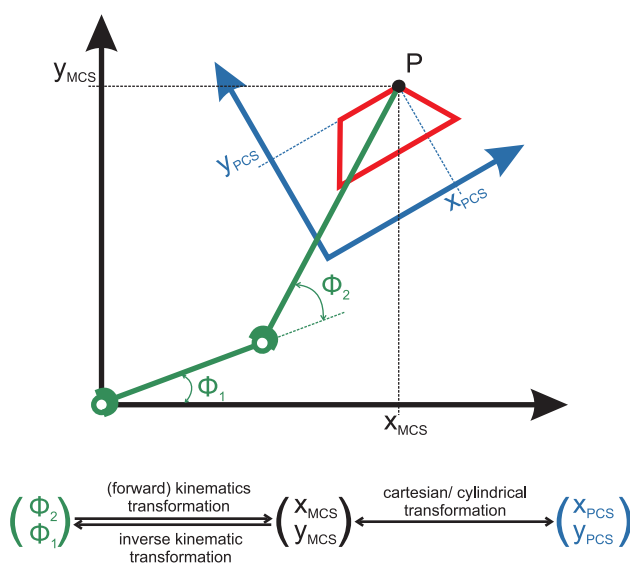
MC_COORD – Koordinované řízení pohybu

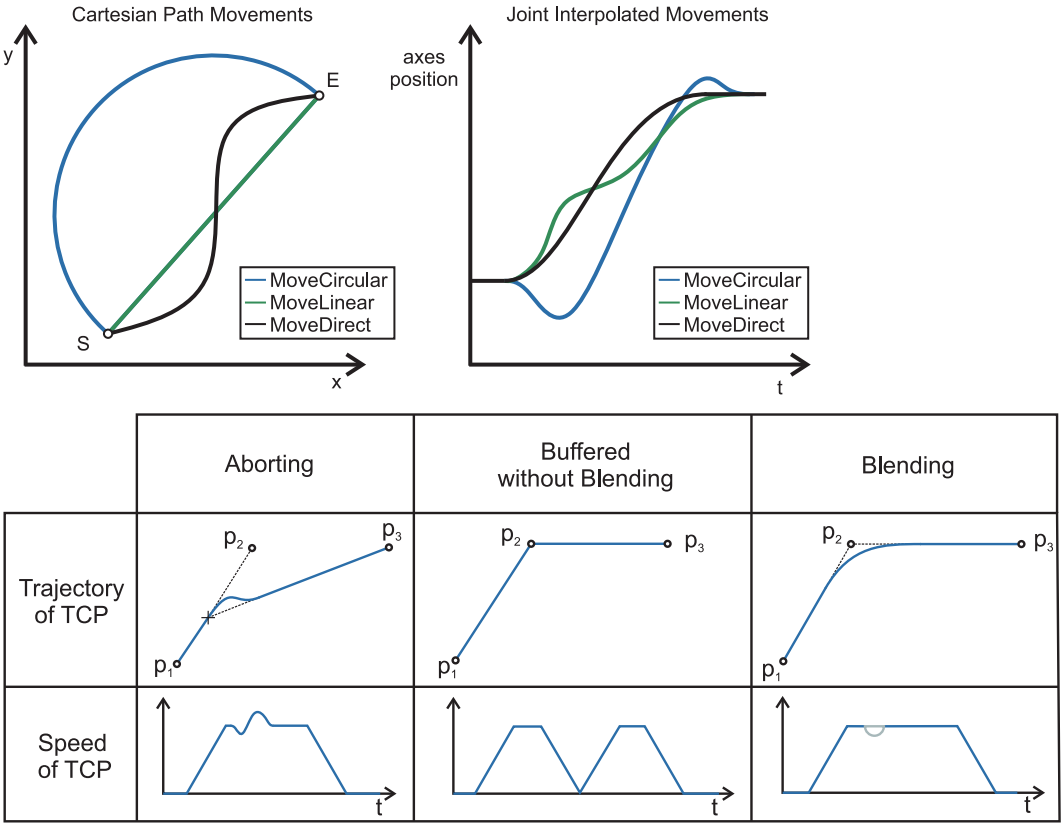
Obsah

MCP_GroupHalt – * Zastavení koordinovaného pohybu (přerušitelné)	805
MCP_GroupInterrupt – * Přerušení pohybu skupiny os	806
MCP_GroupSetOverride – * Nastavení násobivých faktorů na osách ve skupině	807
MCP_GroupSetPosition – * Nastavení polohového offsetu skupiny os	808
MCP_GroupStop – * Zastavení koordinovaného pohybu	809
MCP_MoveCircularAbsolute – * Pohyb do pozice po kružnicích (absolutní souřadnice)	810
MCP_MoveCircularRelative – * Pohyb do pozice po kružnicích (relativní souřadnice)	812
MCP_MoveDirectAbsolute – * Nekoordinovaný pohyb do pozice (absolutní souřadnice)	814
MCP_MoveDirectRelative – * Nekoordinovaný pohyb do pozice (relativní souřadnice)	816
MCP_MoveLinearAbsolute – * Pohyb do pozice po přímkách (absolutní souřadnice)	818
MCP_MoveLinearRelative – * Pohyb do pozice po přímkách (relativní souřadnice)	820
MCP_MovePath – * Generování obecné trajektorie v prostoru . . .	822
MCP_MovePath_PH – * Generování obecné trajektorie v prostoru PH	824
MCP_SetCartesianTransform – * Kartézské transformace	826
MCP_SetKinTransform_Arm – * Kinematické transformace robot ARM	827
MCP_SetKinTransform_UR – * Kinematická transformace pro UR robot	829
MC_AddAxisToGroup – Přidání osy do skupiny os	831
MC_GroupContinue – Pokračování v přerušeném pohybu	832

MC_GroupDisable – Převedení skupiny do stavu GroupDisabled . . .	833
MC_GroupEnable – Převedení skupiny do stavu GroupStandby . . .	834
MC_GroupHalt, MCP_GroupHalt – Zastavení koordinovaného pohybu (přerušitelné)	835
MC_GroupInterrupt, MCP_GroupInterrupt – Přerušení pohybu sku- piny os	841
MC_GroupReadActualAcceleration – Aktuální zrychlení skupiny os	842
MC_GroupReadActualPosition – Aktuální poloha skupiny os	843
MC_GroupReadActualVelocity – Aktuální rychlost skupiny os . . .	844
MC_GroupReadError – Chyby ve skupině os	845
MC_GroupReadStatus – Stav skupin os	846
MC_GroupReset – Nulování chyb os ve skupině	848
MC_GroupSetOverride, MCP_GroupSetOverride – Nastavení násobi- vých faktorů na osách ve skupině	849
MC_GroupSetPosition, MCP_GroupSetPosition – Nastavení poloho- vého offsetu skupiny os	852
MC_GroupStop, MCP_GroupStop – Zastavení koordinovaného pohybu	854
MC_MoveCircularAbsolute, MCP_MoveCircularAbsolute – Pohyb do pozice po kružnicích (absolutní souřadnice)	857
MC_MoveCircularRelative, MCP_MoveCircularRelative – Pohyb do pozice po kružnicích (relativní souřadnice)	862
MC_MoveDirectAbsolute, MCP_MoveDirectAbsolute – Nekoordinovaný pohyb do pozice (absolutní souřadnice)	867
MC_MoveDirectRelative, MCP_MoveDirectRelative – Nekoordinovaný pohyb do pozice (relativní souřadnice)	871
MC_MoveLinearAbsolute, MCP_MoveLinearAbsolute – Pohyb do po- zice po přímkách (absolutní souřadnice)	875
MC_MoveLinearRelative, MCP_MoveLinearRelative – Pohyb do po- zice po přímkách (relativní souřadnice)	879
MC_MovePath, MCP_MovePath – Generování obecné trajektorie v pro- storu	883
MC_MovePath_PH – * Generování obecné trajektorie v prostoru PH	886
MC_ReadCartesianTransform – Přečtení použité kartézské transfor- mace	888
MC_SetCartesianTransform, MCP_SetCartesianTransform – Kartéz- ská transformace	890
MC_UngroupAllAxes – Odebrání všech os ze skupiny	893
RM_AxesGroup – Skupina os pro koordinované řízení pohybu . . .	894
RM_Feed – MC „krmič“	899
RM_Gcode – CNC řízení pohybu	901
RM_GroupTrack – * Sledování polohy/rychlosti	904

Knihovna MC_COORD je speciálně navržena pro koordinaci víceosého řízení pohybu v rámci komplexních systémů. Zahrnuje řadu bloků, včetně **MC_MoveLinearAbsolute** pro provádění přesných lineárních pohybů, doplněných o **MC_MoveLinearRelative** pro relativní lineární pohyb. Pro provádění kruhového pohybu knihovna obsahuje bloky **MC_MoveCircularAbsolute** a **MC_MoveCircularRelative**, což zajišťuje podrobné kruhové trajektorie. V kontextu správy řízení skupinových os tato knihovna zavádí **MC_AddAxisToGroup**, který je dále podporován funkcemi jako **MC_GroupEnable** pro aktivaci, **MC_GroupDisable** pro deaktivaci a **MC_GroupHalt** pro okamžité zastavení seskupených os. Dále knihovna poskytuje **MC_MoveDirectAbsolute** a **MC_MoveDirectRelative**, které umožňují přímou kontrolu nad pohyby os. Pro navigaci po složitých cestách je k dispozici **MC_MovePath**. Základní funkce monitorování a ovládání jsou usnadněny pomocí **MC_GroupReadActualPosition** pro údaje o poloze, **MC_GroupReadActualVelocity** pro přehled rychlosti, **MC_GroupReadError** pro detekci chyb a **MC_GroupReadStatus** pro aktualizace stavu. Knihovna navíc integruje **MC_ReadCartesianTransform** a **MC_SetCartesianTransform**, které jsou životně důležité pro procesy karteziánské transformace. Tato kolekce funkcí podtrhuje význam knihovny v aplikacích, které vyžadují synchronizované řízení více os, zejména v oblastech robotiky a automatizačních systémů.

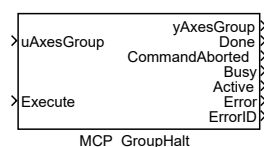




MCP_GroupHalt — * Zastavení koordinovaného pohybu (přerušitelné)

Symbol bloku

Licence: [COORDINATED MOTION](#)



Popis funkce

Popis tohoto bloku ještě není k dispozici. Níže naleznete částečný popis vstupů, výstupů a parametrů bloku. Kompletní popis bloku bude k dispozici v dalších revizích dokumentace.

Tento blok nepropaguje kvalitu signálu. Více informací je uvedeno v sekci [1.4](#).

Vstup

uAxesGroup	Odkaz na skupinu os	Reference
Execute	Náběžná hrana aktivuje blok	Bool

Parametr

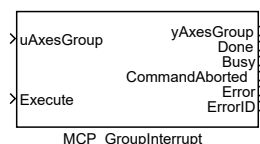
Deceleration	Maximální povolené zpomalení [unit/s ²]	Double (F64)
Jerk	Maximální povolená změna zrychlení [unit/s ³]	Double (F64)
BufferMode	Režim převzetí osy	⊙1 Long (I32)
	1 aborting	
	2 buffered	
LimitMode	Význam limitů rychlosti/zrychlení/změny zrychlení	⊙1 Long (I32)
	1 Relativní (k defaultním hodnotám)	
	2 Absolutní [unit/s unit/s ² ...]	
Superimposed	příznak spuštění jako aditivní pohyb k existujícímu	Bool

Výstup

yAxesGroup	Odkaz na skupinu os	Reference
Done	Příznak dokončení algoritmu	Bool
CommandAborted	Příznak přerušení funkce bloku	Bool
Busy	Příznak, že algoritmus ještě neskončil	Bool
Active	Příznak, že blok řídí osu	Bool
Error	Příznak chyby	Bool
ErrorID	Výsledek poslední operace	Error
	i kód chyby REXYGENu	

MCP_GroupInterrupt – * Přerušení pohybu skupiny os

Symbol bloku

Licence: [COORDINATED MOTION](#)

Popis funkce

Popis tohoto bloku ještě není k dispozici. Níže naleznete částečný popis vstupů, výstupů a parametrů bloku. Kompletní popis bloku bude k dispozici v dalších revizích dokumentace.

Tento blok nepropaguje kvalitu signálu. Více informací je uvedeno v sekci [1.4](#).

Vstup

uAxesGroup	Odkaz na skupinu os	Reference
Execute	Náběžná hrana aktivuje blok	Bool

Parametr

Deceleration	Maximální povolené zpomalení [unit/s ²]	Double (F64)
Jerk	Maximální povolená změna zrychlení [unit/s ³]	Double (F64)
LimitMode	Význam limitů rychlosti/zrychlení/změny zrychlení	⊙1 Long (I32)
	1 Relativní (k defaultním hodnotám)	
	2 Absolutní [unit/s unit/s ² ...]	

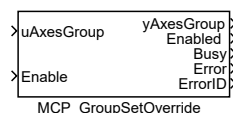
Výstup

yAxesGroup	Odkaz na skupinu os	Reference
Done	Příznak dokončení algoritmu	Bool
Busy	Příznak, že algoritmus ještě neskončil	Bool
CommandAborted	Příznak přerušení funkce bloku	Bool
Error	Příznak chyby	Bool
ErrorID	Výsledek poslední operace	Error
	i kód chyby REXYGENu	

MCP_GroupSetOverride – * Nastavení násobivých faktorů na osách ve skupině

Symbol bloku

Licence: [COORDINATED MOTION](#)



Popis funkce

Popis tohoto bloku ještě není k dispozici. Níže naleznete částečný popis vstupů, výstupů a parametrů bloku. Kompletní popis bloku bude k dispozici v dalších revizích dokumentace.

Tento blok nepropaguje kvalitu signálu. Více informací je uvedeno v sekci [1.4](#).

Vstup

uAxesGroup	Odkaz na skupinu os	Reference
Enable	Povolení funkce bloku (aktivace výstupů)	Bool

Parametr

diff	Pásmo necitlivosti (pro přepočet trajektorie)	⊙0.05	Double (F64)
VelFactor	Faktor násobení pro rychlost	⊙1.0	Double (F64)
AccFactor	Faktor násobení pro zrychlení	⊙1.0	Double (F64)
JerkFactor	Faktor násobení pro změnu zrychlení	⊙1.0	Double (F64)

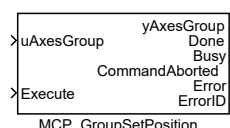
Výstup

yAxesGroup	Odkaz na skupinu os	Reference
Enabled	Signalizuje úspěšné nastavení násobivých faktorů	Bool
Busy	Příznak, že algoritmus ještě neskončil	Bool
Error	Příznak chyby	Bool
ErrorID	Výsledek poslední operace	Error
	i kód chyby REXYGENu	

MCP_GroupSetPosition – * Nastavení polohového offsetu skupiny os

Symbol bloku

Licence: [COORDINATED MOTION](#)



Popis funkce

Popis tohoto bloku ještě není k dispozici. Níže naleznete částečný popis vstupů, výstupů a parametrů bloku. Kompletní popis bloku bude k dispozici v dalších revizích dokumentace.

Tento blok nepropaguje kvalitu signálu. Více informací je uvedeno v sekci [1.4](#).

Vstup

uAxesGroup	Odkaz na skupinu os	Reference
Execute	Náběžná hrana aktivuje blok	Bool

Parametr

Relative	Výběr absolutních (=false) nebo relativních (=true) souřadnic	Bool
CoordSystem	Volba souřadného systému	⊙3 Long (I32)
	1 ACS	
	2 MCS	
	3 PCS	
	4 TCS	
Position_	Pole souřadnic (pozic a orientací)	Double (F64)
	⊙[0.0 0.0 0.0 0.0 0.0 0.0]	

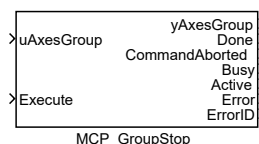
Výstup

yAxesGroup	Odkaz na skupinu os	Reference
Done	Příznak dokončení algoritmu	Bool
Busy	Příznak, že algoritmus ještě neskončil	Bool
CommandAborted	Příznak přerušení funkce bloku	Bool
Error	Příznak chyby	Bool
ErrorID	Výsledek poslední operace	Error
	i kód chyby REXYGENu	

MCP_GroupStop – * Zastavení koordinovaného pohybu

Symbol bloku

Licence: [COORDINATED MOTION](#)



Popis funkce

Popis tohoto bloku ještě není k dispozici. Níže naleznete částečný popis vstupů, výstupů a parametrů bloku. Kompletní popis bloku bude k dispozici v dalších revizích dokumentace.

Tento blok nepropaguje kvalitu signálu. Více informací je uvedeno v sekci [1.4](#).

Vstup

uAxesGroup	Odkaz na skupinu os	Reference
Execute	Náběžná hrana aktivuje blok	Bool

Parametr

Deceleration	Maximální povolené zpomalení [unit/s ²]	Double (F64)
Jerk	Maximální povolená změna zrychlení [unit/s ³]	Double (F64)

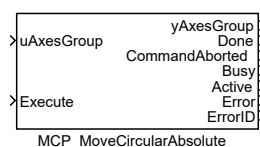
Výstup

yAxesGroup	Odkaz na skupinu os	Reference
Done	Příznak dokončení algoritmu	Bool
CommandAborted	Příznak přerušení funkce bloku	Bool
Busy	Příznak, že algoritmus ještě neskončil	Bool
Active	Příznak, že blok řídí osu	Bool
Error	Příznak chyby	Bool
ErrorID	Výsledek poslední operace	Error
	i kód chyby REXYGENu	

MCP_MoveCircularAbsolute – * Pohyb do pozice po kružnicích (absolutní souřadnice)

Symbol bloku

Licence: [COORDINATED MOTION](#)



Popis funkce

Popis tohoto bloku ještě není k dispozici. Níže naleznete částečný popis vstupů, výstupů a parametrů bloku. Kompletní popis bloku bude k dispozici v dalších revizích dokumentace.

Tento blok nepropaguje kvalitu signálu. Více informací je uvedeno v sekci [1.4](#).

Vstup

uAxesGroup	Odkaz na skupinu os	Reference
Execute	Náběžná hrana aktivuje blok	Bool

Parametr

CircMode	Určuje význam vstupních signálů AuxPoint a CircDirection	Long (I32)
	1 BORDER 2 CENTER 3 RADIUS	⊙1
PathChoice	Volba směru	⊙1 Long (I32)
	1 Ve směru hodinových ručiček 2 Protisměru hodinových ručiček	
Velocity	Maximální povolená rychlost [unit/s]	Double (F64)
Acceleration	Maximální povolené zrychlení [unit/s^2]	Double (F64)
Jerk	Maximální povolená změna zrychlení [unit/s^3]	Double (F64)
CoordSystem	Volba souřadného systému	⊙1 Long (I32)
	1 ACS 2 MCS 3 PCS 4 TCS	

BufferMode	Režim převzetí osy	⊙1	Long (I32)
	1 aborting		
	2 buffered		
	3 blending low		
	4 blending high		
	5 blending previous		
	6 blending next		
LimitMode	Význam limitů rychlosti/zrychlení/změny zrychlení	⊙1	Long (I32)
	1 Relativní (k defaultním hodnotám)		
	2 Absolutní [unit/s unit/s^2 ...]		
TransitionMode	Režim míchání pohybu	⊙1	Long (I32)
	1 TMNone		
	2 TMStartVelocity		
	3 TMConstantVelocity		
	4 TMCornerDistance		
	5 TMMaxCornerDeviation		
	11 Smooth		
TransitionParameter	Parametr pro navázání pohybu (dle zvoleného režimu míchání)		Double (F64)
Superimposed	příznak spuštění jako aditivní pohyb k existujícímu		Bool
AuxPoint	Absolutně zadaná pozice	⊙[0.0 0.0 0.0 0.0 0.0 0.0]	Double (F64)
EndPoint	Absolutně zadaná pozice koncového bodu	⊙[0.0 0.0 0.0 0.0 0.0 0.0]	Double (F64)

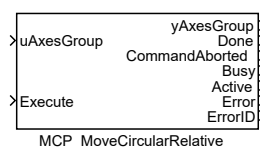
Výstup

yAxesGroup	Odkaz na skupinu os	Reference
Done	Příznak dokončení algoritmu	Bool
CommandAborted	Příznak přerušení funkce bloku	Bool
Busy	Příznak, že algoritmus ještě neskončil	Bool
Active	Příznak, že blok řídí osu	Bool
Error	Příznak chyby	Bool
ErrorID	Výsledek poslední operace	Error
	i kód chyby REXYGENu	

MCP_MoveCircularRelative – * Pohyb do pozice po kružnicích (relativní souřadnice)

Symbol bloku

Licence: [COORDINATED MOTION](#)



Popis funkce

Popis tohoto bloku ještě není k dispozici. Níže naleznete částečný popis vstupů, výstupů a parametrů bloku. Kompletní popis bloku bude k dispozici v dalších revizích dokumentace.

Tento blok nepropaguje kvalitu signálu. Více informací je uvedeno v sekci [1.4](#).

Vstup

uAxesGroup	Odkaz na skupinu os	Reference
Execute	Náběžná hrana aktivuje blok	Bool

Parametr

CircMode	Určuje význam vstupních signálů AuxPoint a CircDirection	Long (I32)
	1 BORDER 2 CENTER 3 RADIUS	⊙1
PathChoice	Volba směru	⊙1 Long (I32)
	1 Ve směru hodinových ručiček 2 Protisměru hodinových ručiček	
Velocity	Maximální povolená rychlost [unit/s]	Double (F64)
Acceleration	Maximální povolené zrychlení [unit/s^2]	Double (F64)
Jerk	Maximální povolená změna zrychlení [unit/s^3]	Double (F64)
CoordSystem	Volba souřadného systému	⊙1 Long (I32)
	1 ACS 2 MCS 3 PCS 4 TCS	

BufferMode	Režim převzetí osy	⊙1	Long (I32)
	1 aborting		
	2 buffered		
	3 blending low		
	4 blending high		
	5 blending previous		
	6 blending next		
LimitMode	Význam limitů rychlosti/zrychlení/změny zrychlení	⊙1	Long (I32)
	1 Relativní (k defaultním hodnotám)		
	2 Absolutní [unit/s unit/s^2 ...]		
TransitionMode	Režim míchání pohybu	⊙1	Long (I32)
	1 TMNone		
	2 TMStartVelocity		
	3 TMConstantVelocity		
	4 TMCornerDistance		
	5 TMMaxCornerDeviation		
	11 Smooth		
TransitionParameter	Parametr pro navázání pohybu (dle zvoleného režimu míchání)		Double (F64)
Superimposed	příznak spuštění jako aditivní pohyb k existujícímu		Bool
AuxPoint	Absolutně zadaná pozice	⊙[0.0 0.0 0.0 0.0 0.0 0.0]	Double (F64)
EndPoint	Absolutně zadaná pozice koncového bodu	⊙[0.0 0.0 0.0 0.0 0.0 0.0]	Double (F64)

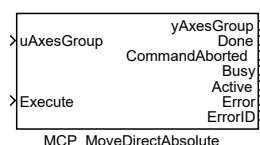
Výstup

yAxesGroup	Odkaz na skupinu os	Reference
Done	Příznak dokončení algoritmu	Bool
CommandAborted	Příznak přerušení funkce bloku	Bool
Busy	Příznak, že algoritmus ještě neskončil	Bool
Active	Příznak, že blok řídí osu	Bool
Error	Příznak chyby	Bool
ErrorID	Výsledek poslední operace	Error
	i kód chyby REXYGENu	

MCP_MoveDirectAbsolute – * Nekoordinovaný pohyb do pozice (absolutní souřadnice)

Symbol bloku

Licence: [COORDINATED MOTION](#)



Popis funkce

Popis tohoto bloku ještě není k dispozici. Níže naleznete částečný popis vstupů, výstupů a parametrů bloku. Kompletní popis bloku bude k dispozici v dalších revizích dokumentace.

Tento blok nepropaguje kvalitu signálu. Více informací je uvedeno v sekci [1.4](#).

Vstup

uAxesGroup	Odkaz na skupinu os	Reference
Execute	Náběžná hrana aktivuje blok	Bool

Parametr

CoordSystem	Volba souřadného systému	⊙1	Long (I32)
1	 ACS		
2	 MCS		
3	 PCS		
4	 TCS		
BufferMode	Režim převzetí osy	⊙1	Long (I32)
1	 aborting		
2	 buffered		
3	 blending low		
4	 blending high		
5	 blending previous		
6	 blending next		
TransitionMode	Režim míchání pohybu	⊙1	Long (I32)
1	 TMNone		
2	 TMStartVelocity		
3	 TMConstantVelocity		
4	 TMCornerDistance		
5	 TMMaxCornerDeviation		
11	 Smooth		
TransitionParameter	Parametr pro navázání pohybu (dle zvoleného režimu míchání)		Double (F64)

Superimposed	příznak spuštění jako aditivní pohyb k existujícímu	Bool
Position_	Pole souřadnic (pozic a orientací) ⊙[0.0 0.0 0.0 0.0 0.0 0.0]	Double (F64)

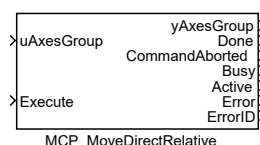
Výstup

yAxesGroup	Odkaz na skupinu os	Reference
Done	Příznak dokončení algoritmu	Bool
CommandAborted	Příznak přerušení funkce bloku	Bool
Busy	Příznak, že algoritmus ještě neskončil	Bool
Active	Příznak, že blok řídí osu	Bool
Error	Příznak chyby	Bool
ErrorID	Výsledek poslední operace i kód chyby REXYGENu	Error

MCP_MoveDirectRelative – * Nekoordinovaný pohyb do pozice (relativní souřadnice)

Symbol bloku

Licence: [COORDINATED MOTION](#)



Popis funkce

Popis tohoto bloku ještě není k dispozici. Níže naleznete částečný popis vstupů, výstupů a parametrů bloku. Kompletní popis bloku bude k dispozici v dalších revizích dokumentace.

Tento blok nepropaguje kvalitu signálu. Více informací je uvedeno v sekci [1.4](#).

Vstup

uAxesGroup	Odkaz na skupinu os	Reference
Execute	Náběžná hrana aktivuje blok	Bool

Parametr

CoordSystem	Volba souřadného systému	⊙1	Long (I32)
1	 ACS		
2	 MCS		
3	 PCS		
4	 TCS		
BufferMode	Režim převzetí osy	⊙1	Long (I32)
1	 aborting		
2	 buffered		
3	 blending low		
4	 blending high		
5	 blending previous		
6	 blending next		
TransitionMode	Režim míchání pohybu	⊙1	Long (I32)
1	 TMNone		
2	 TMStartVelocity		
3	 TMConstantVelocity		
4	 TMCornerDistance		
5	 TMMaxCornerDeviation		
11	 Smooth		
TransitionParameter	Parametr pro navázání pohybu (dle zvoleného režimu míchání)		Double (F64)

Superimposed	příznak spuštění jako aditivní pohyb k existujícímu	Bool
Distance	Pole souřadnic (relativních pozic a orientací) ⊙[0.0 0.0 0.0 0.0 0.0 0.0]	Double (F64)

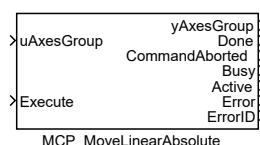
Výstup

yAxesGroup	Odkaz na skupinu os	Reference
Done	Příznak dokončení algoritmu	Bool
CommandAborted	Příznak přerušení funkce bloku	Bool
Busy	Příznak, že algoritmus ještě neskončil	Bool
Active	Příznak, že blok řídí osu	Bool
Error	Příznak chyby	Bool
ErrorID	Výsledek poslední operace i kód chyby REXYGENu	Error

MCP_MoveLinearAbsolute – * Pohyb do pozice po přímkách (absolutní souřadnice)

Symbol bloku

Licence: [COORDINATED MOTION](#)



Popis funkce

Popis tohoto bloku ještě není k dispozici. Níže naleznete částečný popis vstupů, výstupů a parametrů bloku. Kompletní popis bloku bude k dispozici v dalších revizích dokumentace.

Tento blok nepropaguje kvalitu signálu. Více informací je uvedeno v sekci [1.4](#).

Vstup

uAxesGroup	Odkaz na skupinu os	Reference
Execute	Náběžná hrana aktivuje blok	Bool

Parametr

Velocity	Maximální povolená rychlost [unit/s]	Double (F64)
Acceleration	Maximální povolené zrychlení [unit/s ²]	Double (F64)
Jerk	Maximální povolená změna zrychlení [unit/s ³]	Double (F64)
CoordSystem	Volba souřadného systému	⊙1 Long (I32)
	1 ACS	
	2 MCS	
	3 PCS	
	4 TCS	
BufferMode	Režim převzetí osy	⊙1 Long (I32)
	1 aborting	
	2 buffered	
	3 blending low	
	4 blending high	
	5 blending previous	
	6 blending next	
LimitMode	Význam limitů rychlosti/zrychlení/změny zrychlení	⊙1 Long (I32)
	1 Relativní (k defaultním hodnotám)	
	2 Absolutní [unit/s unit/s ² ...]	

TransitionMode	Režim míchání pohybu	⊙1	Long (I32)
1	 TMNone		
2	 TMStartVelocity		
3	 TMConstantVelocity		
4	 TMCornerDistance		
5	 TMMaxCornerDeviation		
11	 Smooth		
TransitionParameter	Parametr pro navázání pohybu (dle zvoleného režimu míchání)		Double (F64)
Superimposed	příznak spuštění jako aditivní pohyb k existujícímu		Bool
Position_	Pole souřadnic (pozic a orientací)		Double (F64)
	⊙[0.0 0.0 0.0 0.0 0.0 0.0]		

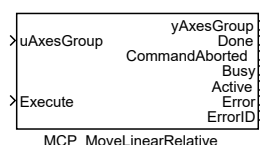
Výstup

yAxesGroup	Odkaz na skupinu os	Reference
Done	Příznak dokončení algoritmu	Bool
CommandAborted	Příznak přerušení funkce bloku	Bool
Busy	Příznak, že algoritmus ještě neskončil	Bool
Active	Příznak, že blok řídí osu	Bool
Error	Příznak chyby	Bool
ErrorID	Výsledek poslední operace	Error
i	 kód chyby REXYGENu	

MCP_MoveLinearRelative – * Pohyb do pozice po přímkách (relativní souřadnice)

Symbol bloku

Licence: [COORDINATED MOTION](#)



Popis funkce

Popis tohoto bloku ještě není k dispozici. Níže naleznete částečný popis vstupů, výstupů a parametrů bloku. Kompletní popis bloku bude k dispozici v dalších revizích dokumentace.

Tento blok nepropaguje kvalitu signálu. Více informací je uvedeno v sekci [1.4](#).

Vstup

uAxesGroup	Odkaz na skupinu os	Reference
Execute	Náběžná hrana aktivuje blok	Bool

Parametr

Velocity	Maximální povolená rychlost [unit/s]	Double (F64)
Acceleration	Maximální povolené zrychlení [unit/s ²]	Double (F64)
Jerk	Maximální povolená změna zrychlení [unit/s ³]	Double (F64)
CoordSystem	Volba souřadného systému	⊙1 Long (I32)
	1 ACS	
	2 MCS	
	3 PCS	
	4 TCS	
BufferMode	Režim převzetí osy	⊙1 Long (I32)
	1 aborting	
	2 buffered	
	3 blending low	
	4 blending high	
	5 blending previous	
	6 blending next	
LimitMode	Význam limitů rychlosti/zrychlení/změny zrychlení	⊙1 Long (I32)
	1 Relativní (k defaultním hodnotám)	
	2 Absolutní [unit/s unit/s ² ...]	

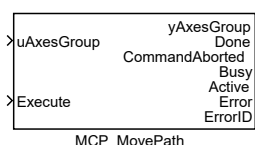
TransitionMode	Režim míchání pohybu	⊙1	Long (I32)
1	 TMNone		
2	 TMStartVelocity		
3	 TMConstantVelocity		
4	 TMCornerDistance		
5	 TMMaxCornerDeviation		
11	 Smooth		
TransitionParameter	Parametr pro navázání pohybu (dle zvoleného režimu míchání)		Double (F64)
Superimposed	příznak spuštění jako aditivní pohyb k existujícímu		Bool
Distance	Pole souřadnic (relativních pozic a orientací)		Double (F64)
	⊙[0.0 0.0 0.0 0.0 0.0 0.0]		

Výstup

yAxesGroup	Odkaz na skupinu os	Reference
Done	Příznak dokončení algoritmu	Bool
CommandAborted	Příznak přerušení funkce bloku	Bool
Busy	Příznak, že algoritmus ještě neskončil	Bool
Active	Příznak, že blok řídí osu	Bool
Error	Příznak chyby	Bool
ErrorID	Výsledek poslední operace	Error
i	 kód chyby REXYGENu	

MCP_MovePath – * **Generování obecné trajektorie v prostoru**

Symbol bloku

Licence: [COORDINATED MOTION](#)

Popis funkce

Popis tohoto bloku ještě není k dispozici. Níže naleznete částečný popis vstupů, výstupů a parametrů bloku. Kompletní popis bloku bude k dispozici v dalších revizích dokumentace.

Tento blok nepropaguje kvalitu signálu. Více informací je uvedeno v sekci [1.4](#).

Vstup

uAxesGroup	Odkaz na skupinu os	Reference
Execute	Náběžná hrana aktivuje blok	Bool

Parametr

TotalTime	Čas celého pohybu	Double (F64)
RampTime	Čas [s] zrychlování/zpomalování	Double (F64)
CoordSystem	Volba souřadného systému	⊙2 Long (I32)
	1 ACS	
	2 MCS	
	3 PCS	
	4 TCS	
BufferMode	Režim převzetí osy	⊙1 Long (I32)
	1 aborting	
	2 buffered	
	3 blending low	
	4 blending high	
	5 blending previous	
	6 blending next	
TransitionMode	Režim míchání pohybu	⊙1 Long (I32)
	1 TMNone	
	2 TMStartVelocity	
	3 TMConstantVelocity	
	4 TMCornerDistance	
	5 TMMaxCornerDeviation	
	11 Smooth	
TransitionParameter	Parametr pro navázání pohybu (dle zvoleného režimu míchání)	Double (F64)

RampIn	RampIn faktor (0 = RampIn režim se nepoužívá)	Double (F64)
Superimposed	příznak spuštění jako aditivní pohyb k existujícímu	Bool
pc	Control-points matrix ⊙[0.0 1.0 2.0; 0.0 1.0 1.0; 0.0 1.0 0.0]	Double (F64)
pk	Knot-points vector ⊙[0.0 0.0 0.0 0.0 0.5 1.0 1.0]	Double (F64)
pw	Váhový vektor ⊙[1.0 1.0 1.0]	Double (F64)
pν	Polynoms for feedrate definition ⊙[0.0 0.05 0.95; 0.0 0.1 0.1; 0.0 0.0 0.0; 0.1 0.0 -0.1; -0.05 0.0 0.05; 0.0 0.0 0.0]	Double (F64)
pt	Knot-points (time [s]) for feedrate ⊙[0.0 1.0 10.0 11.0]	Double (F64)
user	Pouze pro special edit ⊙[0.0 1.0 2.0 3.0]	Double (F64)

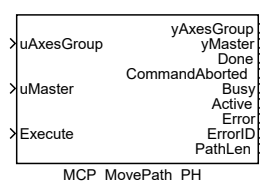
Výstup

yAxesGroup	Odkaz na skupinu os	Reference
Done	Příznak dokončení algoritmu	Bool
CommandAborted	Příznak přerušení funkce bloku	Bool
Busy	Příznak, že algoritmus ještě neskončil	Bool
Active	Příznak, že blok řídí osu	Bool
Error	Příznak chyby	Bool
ErrorID	Výsledek poslední operace i kód chyby REXYGENu	Error

MCP_MovePath_PH – * Generování obecné trajektorie v prostoru PH

Symbol bloku

Licence: [COORDINATED MOTION](#)



Popis funkce

Popis tohoto bloku ještě není k dispozici. Níže naleznete částečný popis vstupů, výstupů a parametrů bloku. Kompletní popis bloku bude k dispozici v dalších revizích dokumentace.

Tento blok nepropaguje kvalitu signálu. Více informací je uvedeno v sekci [1.4](#).

Vstup

uAxesGroup	Odkaz na skupinu os	Reference
uMaster	Odkaz na osu (přípustné je jen spojení RM_Axis.axisRef-uAxis nebo yAxis-uAxis)	Reference
Execute	Náběžná hrana aktivuje blok	Bool

Parametr

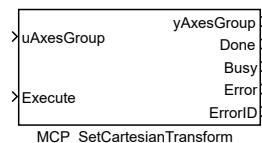
CoordSystem	Volba souřadného systému	⊙2	Long (I32)
	1 ACS		
	2 MCS		
	3 PCS		
	4 TCS		
BufferMode	Režim převzetí osy	⊙1	Long (I32)
	1 aborting		
	2 buffered		
	3 blending low		
	4 blending high		
	5 blending previous		
	6 blending next		
Cyclic	Příznak cyklicky opakovaného profilu		Bool
RampIn	RampIn faktor (0 = RampIn režim se nepoužívá)		Double (F64)
Superimposed	příznak spuštění jako aditivní pohyb k existujícímu		Bool
UZV1		⊙[]	Double (F64)
Points		⊙[]	Double (F64)

Výstup

<code>yAxesGroup</code>	Odkaz na skupinu os	Reference
<code>yMaster</code>	Odkaz na osu (přípustné je jen spojení <code>RM_Axis.axisRef-uAxis</code> nebo <code>yAxis-uAxis</code>)	Reference
<code>Done</code>	Příznak dokončení algoritmu	Bool
<code>CommandAborted</code>	Příznak přerušení funkce bloku	Bool
<code>Busy</code>	Příznak, že algoritmus ještě neskončil	Bool
<code>Active</code>	Příznak, že blok řídí osu	Bool
<code>Error</code>	Příznak chyby	Bool
<code>ErrorID</code>	Výsledek poslední operace	Error
<code>PathLen</code>		Double (F64)

MCP_SetCartesianTransform – * **Kartézské transformace**

Symbol bloku

Licence: **COORDINATED MOTION**

Popis funkce

Popis tohoto bloku ještě není k dispozici. Níže naleznete částečný popis vstupů, výstupů a parametrů bloku. Kompletní popis bloku bude k dispozici v dalších revizích dokumentace.

Tento blok nepropaguje kvalitu signálu. Více informací je uvedeno v sekci [1.4](#).

Vstup

uAxesGroup	Odkaz na skupinu os	Reference
Execute	Náběžná hrana aktivuje blok	Bool

Parametr

SelTrans	Dotčená transformace souřadnic	⊙1	Long (I32)
	1 Základní souřadnice výrobku (PCS)		
	2 Posunutí nástroje		
	3 Souřadnice základny stroje		
TransX	Posun v ose X		Double (F64)
TransY	Posun v ose Y		Double (F64)
TransZ	Posun v ose Z		Double (F64)
RotAngle1	Úhel rotace podél osy Z		Double (F64)
RotAngle2	Úhel rotace podél osy Y		Double (F64)
RotAngle3	Úhel rotace podél osy X		Double (F64)
Relative	Výběr absolutních (=false) nebo relativních (=true) souřadnic		Bool
SSF	xyz		Bool

Výstup

yAxesGroup	Odkaz na skupinu os	Reference
Done	Příznak dokončení algoritmu	Bool
Busy	Příznak, že algoritmus ještě neskončil	Bool
Error	Příznak chyby	Bool
ErrorID	Výsledek poslední operace	Error
	i kód chyby REXYGENu	

MCP_SetKinTransform_Arm – * Kinematické transformace robot ARM

Symbol bloku

Licence: [COORDINATED MOTION](#)



Popis funkce

Popis tohoto bloku ještě není k dispozici. Níže naleznete částečný popis vstupů, výstupů a parametrů bloku. Kompletní popis bloku bude k dispozici v dalších revizích dokumentace.

Tento blok nepropaguje kvalitu signálu. Více informací je uvedeno v sekci [1.4](#).

Vstup

uAxesGroup	Odkaz na skupinu os	Reference
Execute	Náběžná hrana aktivuje blok	Bool

Parametr

arot	Úhel pro jednu otáčku (ACS)	⊙6.28318531	Double (F64)
mrot	Úhel pro jednu otáčku (MCS)	⊙6.28318531	Double (F64)
irt	Způsob zadávání otočení	⊙1	Long (I32)
	1 ZYX úhly		
	2 Katernion		
a1		⊙400.0	Double (F64)
a2		⊙300.0	Double (F64)
a3			Double (F64)
d1			Double (F64)
d23			Double (F64)
d4		⊙200.0	Double (F64)
d6		⊙100.0	Double (F64)
xe			Double (F64)
ye			Double (F64)
ze			Double (F64)
gamaE			Double (F64)
betaE			Double (F64)
alphaE			Double (F64)

Výstup

		Reference
yAxesGroup	Odkaz na skupinu os	
Done	Příznak dokončení algoritmu	Bool
Busy	Příznak, že algoritmus ještě neskončil	Bool
Error	Příznak chyby	Bool
ErrorID	Výsledek poslední operace	Error
	i kód chyby REXYGENu	

MCP_SetKinTransform_UR – * Kinematická transformace pro UR robot

Symbol bloku

Licence: [COORDINATED MOTION](#)



Popis funkce

Popis tohoto bloku ještě není k dispozici. Níže naleznete částečný popis vstupů, výstupů a parametrů bloku. Kompletní popis bloku bude k dispozici v dalších revizích dokumentace.

Tento blok nepropaguje kvalitu signálu. Více informací je uvedeno v sekci [1.4](#).

Vstup

uAxesGroup	Odkaz na skupinu os	Reference
Execute	Náběžná hrana aktivuje blok	Bool

Parametr

arot	Úhel pro jednu otáčku (ACS)	⊙6.28318531	Double (F64)
mrot	Úhel pro jednu otáčku (MCS)	⊙6.28318531	Double (F64)
irt	Způsob zadávání otočení	⊙1	Long (I32)
	1 ZYX úhly		
	2 Katernion		
11			Double (F64)
12			Double (F64)
13			Double (F64)
14		⊙400.0	Double (F64)
15		⊙400.0	Double (F64)
16		⊙400.0	Double (F64)
17		⊙400.0	Double (F64)
18		⊙400.0	Double (F64)

Výstup

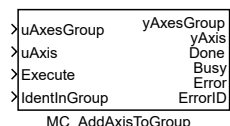
yAxesGroup	Odkaz na skupinu os	Reference
Done	Příznak dokončení algoritmu	Bool
Busy	Příznak, že algoritmus ještě neskončil	Bool
Error	Příznak chyby	Bool

ErrorID	Výsledek poslední operace	Error
i	 kód chyby REXYGENu	

MC_AddAxisToGroup – Přidání osy do skupiny os

Symbol bloku

Licence: [COORDINATED MOTION](#)



Popis funkce

Blok `MC_AddAxisToGroup` přidá do skupiny os `uAxesGroup` osu `uAxis`. Skupina os `uAxesGroup` se založí pomocí bloku [RM_AxesGroup](#). Osa `uAxis` přiváděná na vstup bloku `MC_AddAxisToGroup` musí být definována stejně jako jednotlivá osa [RM_Axis](#) z knihovny `MC_SINGLE`.

Poznámka 1: Každý `IdentInGroup` může být použit pouze jednou, jinak nastane chyba.

Vstupy

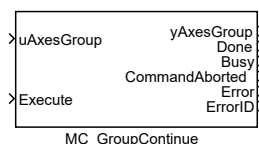
<code>uAxesGroup</code>	Odkaz na skupinu os	Reference
<code>uAxis</code>	Odkaz na osu (přípustné je jen spojení <code>RM_Axis.axisRef-uAxis</code> nebo <code>yAxis-uAxis</code>)	Reference
<code>Execute</code>	Náběžná hrana aktivuje blok	Bool
<code>IdentInGroup</code>	Pořadí osy ve skupině (0=první nepřřazená)	Long (I32)

Výstupy

<code>yAxesGroup</code>	Odkaz na skupinu os	Reference
<code>yAxis</code>	Odkaz na osu (přípustné je jen spojení <code>RM_Axis.axisRef-uAxis</code> nebo <code>yAxis-uAxis</code>)	Reference
<code>Done</code>	Příznak dokončení algoritmu	Bool
<code>Busy</code>	Příznak, že algoritmus ještě neskončil	Bool
<code>Error</code>	Příznak chyby	Bool
<code>ErrorID</code>	Výsledek poslední operace i obecná chyba systému REXYGEN	Error

MC_GroupContinue – Pokračování v přerušeném pohybu

Symbol bloku

Licence: [COORDINATED MOTION](#)

Popis funkce

Blok **MC_GroupContinue** ruší působení bloku **MC_GroupInterrupt**. Skupina os dokončí původně prováděný pohyb před přerušením. Výstup **Done** určuje, zda se pohyb vrátil do stavu před přerušením.

Vstupy

uAxesGroup	Odkaz na skupinu os	Reference
Execute	Náběžná hrana aktivuje blok	Bool

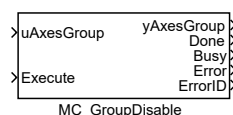
Výstupy

yAxesGroup	Odkaz na skupinu os	Reference
Done	Příznak dokončení algoritmu	Bool
Busy	Příznak, že algoritmus ještě neskončil	Bool
CommandAborted	Příznak přerušení funkce bloku	Bool
Error	Příznak chyby	Bool
ErrorID	Výsledek poslední operace i obecná chyba systému REXYGEN	Error

MC_GroupDisable – Převedení skupiny do stavu GroupDisabled

Symbol bloku

Licence: [COORDINATED MOTION](#)



Popis funkce

Blok **MC_GroupDisable** mění stav skupiny os na stav „GroupDisabled“. Pokud je skupina os v pohybu, tak je nejprve převedena do stavu „Stopping“. Zastavuje se s maximální možnou hodnotou zpomalení (nastavenou v bloku [RM_Axis](#)). Po zastavení je skupina os převedena do stavu „GroupDisabled“ (viz. stavový diagram u bloku [RM_AxesGroup](#)).

Poznámka 1: Příkaz neovlivňuje napájení žádné z os ve skupině os.

Vstupy

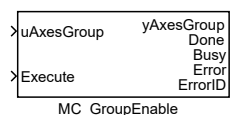
uAxesGroup	Odkaz na skupinu os	Reference
Execute	Náběžná hrana aktivuje blok	Bool

Výstupy

yAxesGroup	Odkaz na skupinu os	Reference
Done	Příznak dokončení algoritmu	Bool
Busy	Příznak, že algoritmus ještě neskončil	Bool
Error	Příznak chyby	Bool
ErrorID	Výsledek poslední operace i obecná chyba systému REXYGEN	Error

MC_GroupEnable – Převedení skupiny do stavu GroupStandby

Symbol bloku

Licence: [COORDINATED MOTION](#)

Popis funkce

Blok **MC_GroupEnable** převede stav skupiny os ze stavu „GroupDisabled“ do stavu „GroupStandby“ (viz. stavový diagram u bloku [RM_AxesGroup](#)).

Poznámka 1: Příkaz neovlivňuje napájení žádné z os ve skupině os.

Vstupy

uAxesGroup	Odkaz na skupinu os	Reference
Execute	Náběžná hrana aktivuje blok	Bool

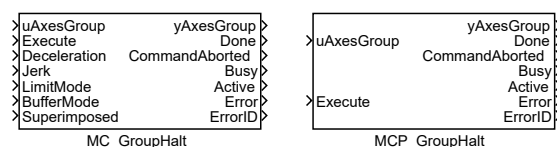
Výstupy

yAxesGroup	Odkaz na skupinu os	Reference
Done	Příznak dokončení algoritmu	Bool
Busy	Příznak, že algoritmus ještě neskončil	Bool
Error	Příznak chyby	Bool
ErrorID	Výsledek poslední operace	Error
	i obecná chyba systému REXYGEN	

MC_GroupHalt, MCP_GroupHalt – Zastavení koordinovaného pohybu (přerušitelné)

Symbyoly bloků

Licence: **COORDINATED MOTION**



Popis funkce

Bloky MC_GroupHalt a MCP_GroupHalt mají naprosto shodnou funkci, jediným rozdílem je, že MCP_ varianta bloku má méně vstupů a potřebné konstanty se zadávají jako parametry bloku.

Blok **MC_GroupHalt** zahajuje řízené zastavení pohybu. Osa se přesune do stavu „GroupMoving“, dokud není rychlost nulová. Společně s nastavením výstupu **Done** je stav změněn na „GroupStandby“.

Poznámka 1: Blok **MC_GroupHalt** se používá k zastavení skupiny os za normálních provozních podmínek. V non-buffered režimu je možné zadat další pohybový příkaz při zpomalení osy, který zruší **MC_GroupHalt** a bude ihned proveden.

Poznámka 2: Je-li tento příkaz aktivní, další příkaz může být aktivován (spuštěn). Např. vozidlo bez řidiče detekuje překážku a potřebuje zastavit. **MC_GroupHalt** je aktivován. Před dosažením stavu „GroupStandby“ je překážka odstraněna a pohyb může pokračovat nastavením dalšího pohybového příkazu, aby vozidlo nemuselo zastavit.

Poznámka 3: Blok nemá parametr **CoordSystem**, protože jej přejímá z předchozího bloku.

Vstupy

uAxesGroup	Odkaz na skupinu os	Reference
Execute	Náběžná hrana aktivuje blok	Bool
Deceleration	Maximální povolené zpomalení [unit/s ²]	Double (F64)
Jerk	Maximální povolená změna zrychlení [unit/s ³]	Double (F64)
LimitMode	Volba jednotek pro limity (Deceleration, Jerk)	⊙1 Long (I32)
1	Relative [part of group limit]	
2	Absolute [unit/s, unit/s ² , unit/s ³]	

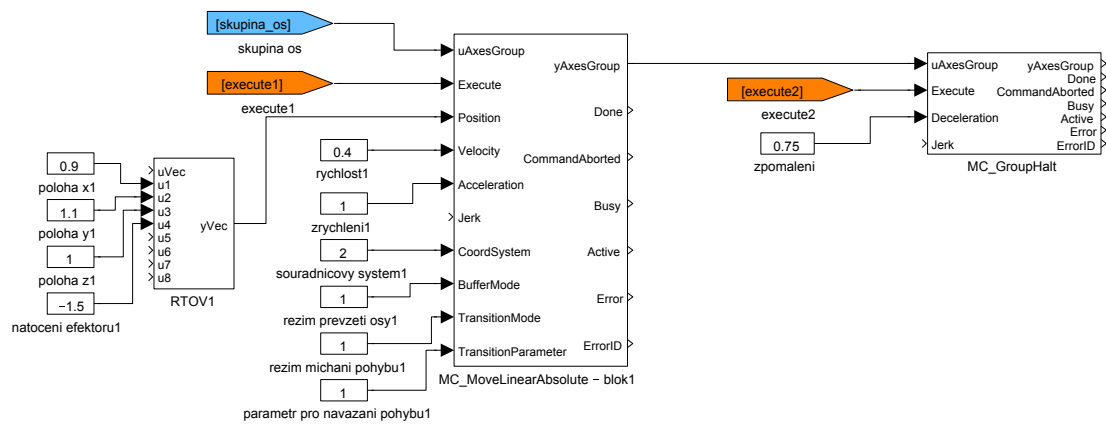
BufferMode	Režim převzetí osy	☉1	Long (I32)
1	 Aborting (nový blok se spustí okamžitě)		
2	 Buffered (nový blok se spustí po dokončení předchozího)		
3	 Blending low (nový blok se spustí po dokončení předchozího, původní pohyb skončí s nižší rychlostí z obou bloků)		
4	 Blending high (nový blok se spustí po dokončení předchozího, původní pohyb skončí s vyšší rychlostí z obou bloků)		
5	 Blending previous (nový blok se spustí po dokončení předchozího, původní pohyb skončí se svojí koncovou rychlostí)		
6	 Blending next (nový blok se spustí po dokončení předchozího, původní pohyb skončí s počáteční rychlostí nového bloku)		
Superimposed	Příznak vykonání jako vedlejší (superimposed) pohyb		Bool

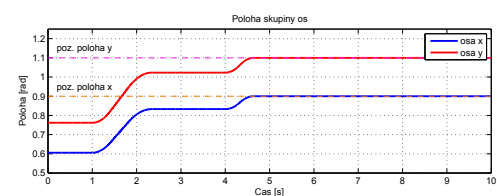
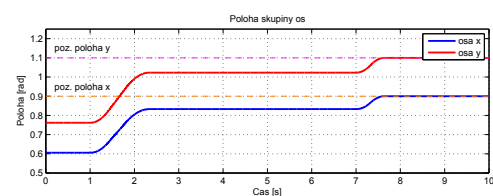
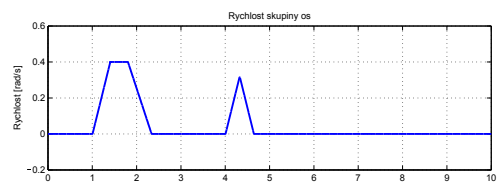
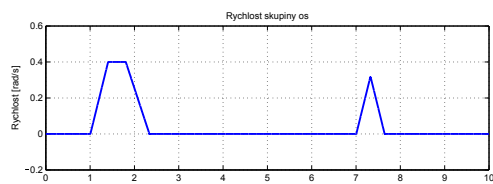
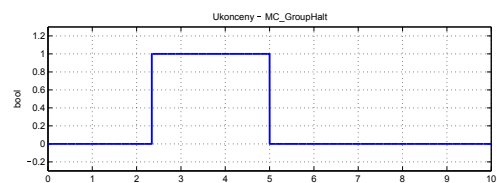
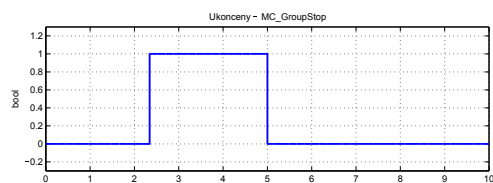
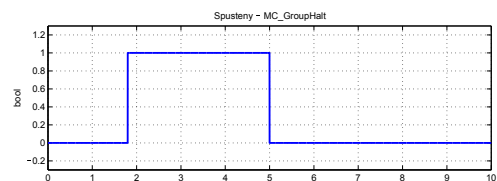
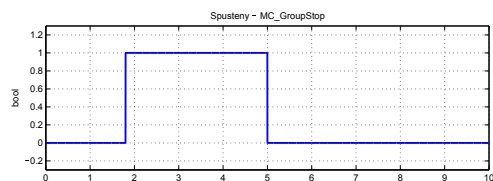
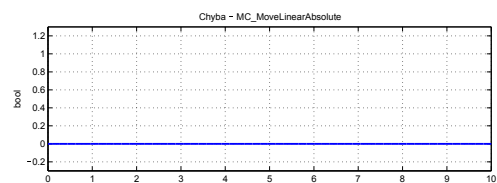
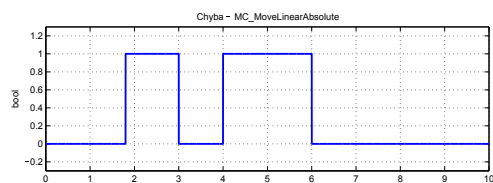
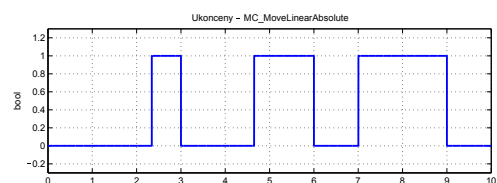
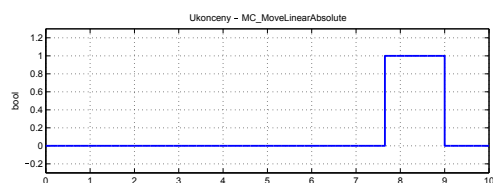
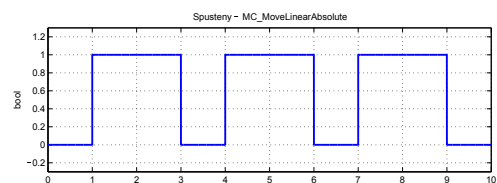
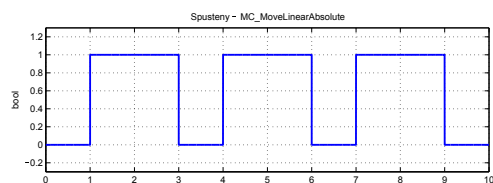
Výstupy

yAxesGroup	Odkaz na skupinu os	Reference
Done	Příznak dokončení algoritmu	Bool
CommandAborted	Příznak přerušení funkce bloku	Bool
Busy	Příznak, že algoritmus ještě neskončil	Bool
Active	Příznak, že blok řídí osu	Bool
Error	Příznak chyby	Bool
ErrorID	Výsledek poslední operace	Error
i	 obecná chyba systému REXYGEN	

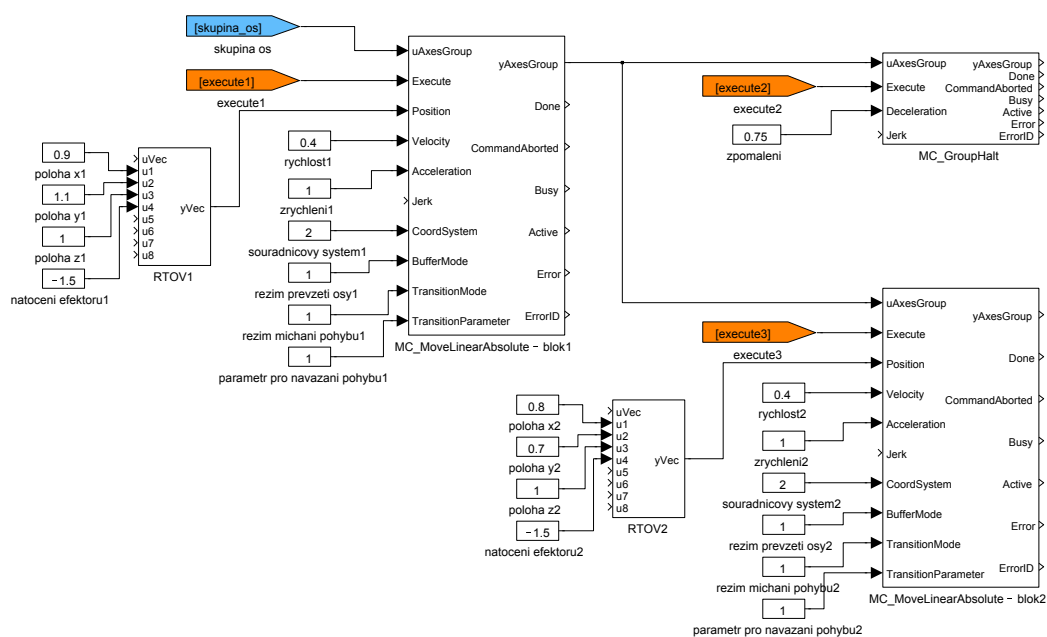
Příklad

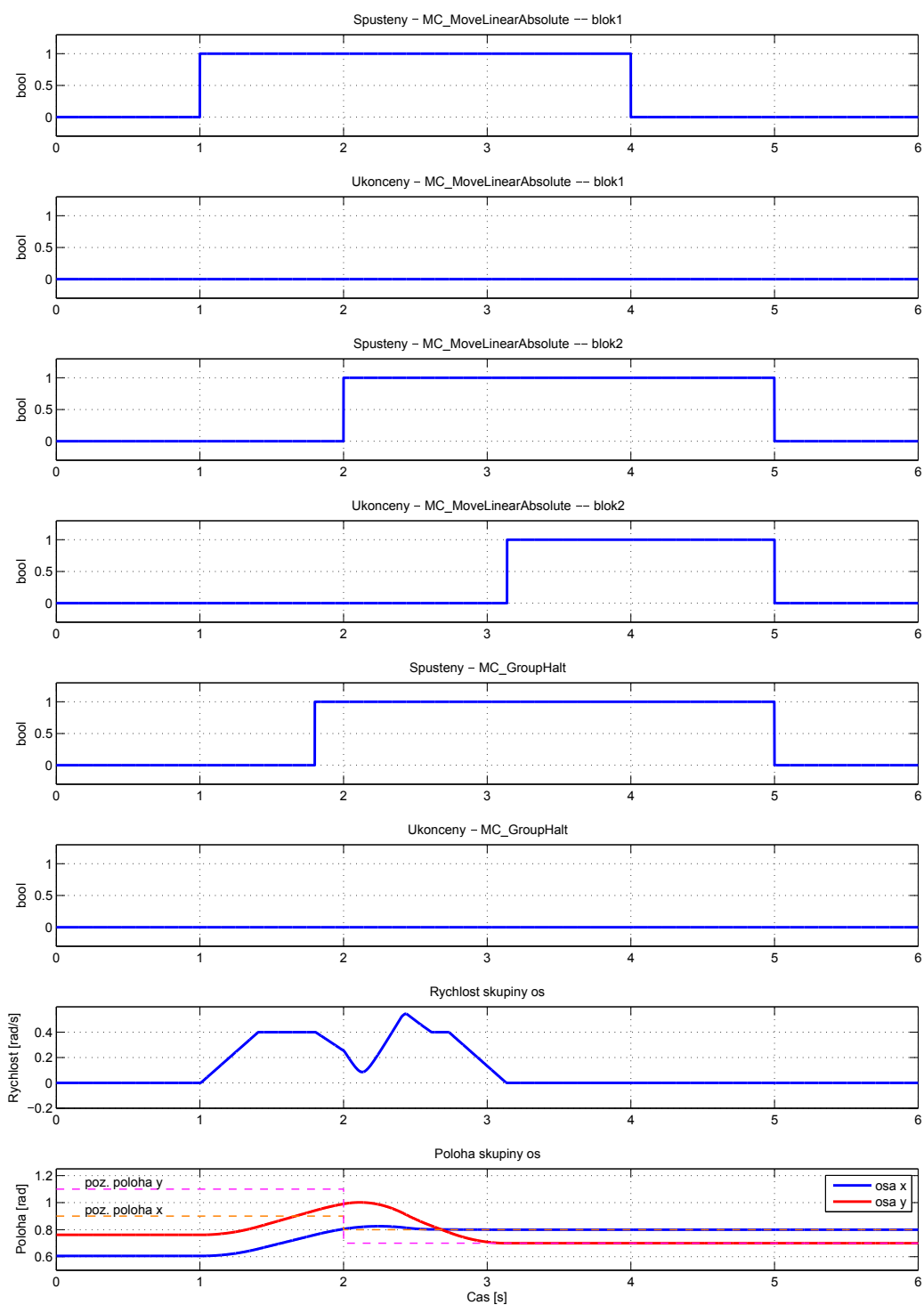
Pro porovnání je následující příklad totožný jako u bloku [MC_GroupStop](#). Časové průběhy v levém sloupci odpovídají bloku [MC_GroupStop](#). Průběhy v pravém sloupci bloku [MC_GroupHalt](#). Je vidět, že při použití bloku [MC_GroupHalt](#) dojde k dojetí do požadované polohy již na druhé spuštění exekutivy bloku [MC_MoveLinearAbsolute](#).





V druhém příkladě je spuštěn blok **MC_MoveLinearAbsolute**, který je následně přerušen blokem **MC_GroupHalt**. Ještě před zastavením je spuštěn druhý pohyb blokem dalším **MC_MoveLinearAbsolute**. Příklad ilustruje to, že pro možnost spuštění dalšího pohybu, nemusí dojít k úplnému zastavení skupiny os.

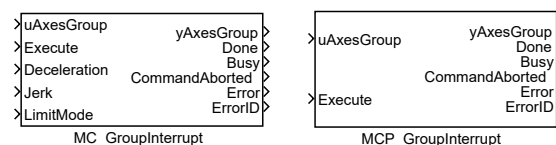




MC_GroupInterrupt, MCP_GroupInterrupt – Přerušení pohybu skupiny os

Symboly bloků

Licence: [COORDINATED MOTION](#)



Popis funkce

Bloky MC_GroupInterrupt a MCP_GroupInterrupt mají naprosto shodnou funkci, jediným rozdílem je, že MCP_ varianta bloku má méně vstupů a potřebné konstanty se zadávají jako parametry bloku.

Blok **MC_GroupInterrupt** přeruší právě prováděný pohyb a uvede skupinu os do klidu, nicméně tento příkaz nezruší přerušený pohyb (výstup **CommandAborted** nebude nastaven na true a **Busy** zůstane true). Informace o původním pohybu zůstane uložena. Skupina os zůstane v původním stavu i po zastavení, kdy je výstup **Done** nastaven na true.

Poznámka 1: Tento blok je spárován s blokem **MC_GroupContinue**, jehož aktivací je skupina os vrácena do stavu, před spuštěním bloku **MC_GroupInterrupt**.

Vstupy

uAxesGroup	Odkaz na skupinu os	Reference
Execute	Náběžná hrana aktivuje blok	Bool
Deceleration	Maximální povolené zpomalení [unit/s ²]	Double (F64)
Jerk	Maximální povolená změna zrychlení [unit/s ³]	Double (F64)
LimitMode	Volba jednotek pro limity (Deceleration, Jerk)	⊙1 Long (I32)
1	Relative [part of group limit]	
2	Absolute [unit/s, unit/s ² , unit/s ³]	

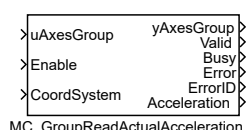
Výstupy

yAxesGroup	Odkaz na skupinu os	Reference
Done	Příznak dokončení algoritmu	Bool
Busy	Příznak, že algoritmus ještě neskončil	Bool
CommandAborted	Příznak přerušení funkce bloku	Bool
Error	Příznak chyby	Bool
ErrorID	Výsledek poslední operace	Error
i	obecná chyba systému REXYGEN	

MC_GroupReadActualAcceleration – Aktuální zrychlení skupiny os

Symbol bloku

Licence: [COORDINATED MOTION](#)



Popis funkce

Blok **MC_GroupReadActualAcceleration** zpřístupňuje na výstup **Position** aktuální zrychlení skupiny os ve zvoleném souřadnicovém systému. Hodnota je platná jen pokud je na výstupu **Valid** true, čehož se dosáhne nastavením vstupu **Enable** na hodnotu true.

Výstupní vektor vždy obsahuje 3 polohové souřadnice (pokud kinematická transformace některé nepoužívá, tak mají hodnotu 0), orientaci ve formě eulerovo úhlů ZYX, (jen pokud ji kinematická transformace používá), dodatečné osy (pokud je kinematická transformace používá). Eulerovo úhly a natočení dodatečných rotačních os je v jednotkách definovaných kinematickou transformací (základní jednotka jsou radiány za sekundu za sekundu).

Vstupy

uAxesGroup	Odkaz na skupinu os	Reference
Enable	Povolení funkce bloku (aktivace výstupů)	Bool
CoordSystem	Volba souřadného systému	Long (I32)
	1 ACM	
	2 MCS	
	3 PCS	

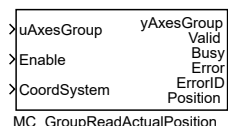
Výstupy

yAxesGroup	Odkaz na skupinu os	Reference
Valid	Příznak platnosti výstupní hodnoty	Bool
Busy	Příznak, že algoritmus ještě neskončil	Bool
Error	Příznak chyby	Bool
ErrorID	Výsledek poslední operace	Error
	i obecná chyba systému REXYGEN	
Acceleration	Pole aktuálních zrychlení skupiny os	Reference

MC_GroupReadActualPosition – Aktuální poloha skupiny os

Symbol bloku

Licence: [COORDINATED MOTION](#)



Popis funkce

Blok `MC_GroupReadActualPosition` zpřístupňuje na výstup `Position` aktuální polohu skupiny os ve zvoleném souřadnicovém systému. Hodnota je platná jen pokud je na výstupu `Valid` true, čehož se dosáhne nastavením vstupu `Enable` na hodnotu true.

Výstupní vektor vždy obsahuje 3 polohové souřadnice (pokud kinematická transformace některé nepoužívá, tak mají hodnotu 0), orientaci ve formě eulerovo úhlů ZYX, (jen pokud ji kinematická transformace používá), dodatečné osy (pokud je kinematická transformace používá). Eulerovo úhly a natočení dodatečných rotačních os je v jednotkách definovaných kinematickou transformací (základní jednotka jsou radiány).

Vstupy

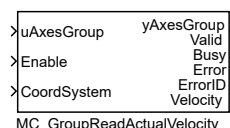
<code>uAxesGroup</code>	Odkaz na skupinu os	Reference
<code>Enable</code>	Povolení funkce bloku (aktivace výstupů)	Bool
<code>CoordSystem</code>	Volba souřadného systému	Long (I32)
	1 ACM	
	2 MCS	
	3 PCS	

Výstupy

<code>yAxesGroup</code>	Odkaz na skupinu os	Reference
<code>Valid</code>	Příznak platnosti výstupní hodnoty	Bool
<code>Busy</code>	Příznak, že algoritmus ještě neskončil	Bool
<code>Error</code>	Příznak chyby	Bool
<code>ErrorID</code>	Výsledek poslední operace	Error
	i obecná chyba systému REXYGEN	
<code>Position</code>	Pole aktuálních souřadnic (pozice a orientace) skupiny os	Reference

MC_GroupReadActualVelocity – Aktuální rychlost skupiny os

Symbol bloku

Licence: [COORDINATED MOTION](#)

Popis funkce

Blok `MC_GroupReadActualVelocity` zpřístupňuje na výstup `Velocity` aktuální rychlost skupiny os ve zvoleném souřadnicovém systému. Hodnota je platná jen pokud je na výstupu `Valid` true, čehož se dosáhne nastavením vstupu `Enable` na hodnotu true.

Výstupní vektor vždy obsahuje 3 polohové souřadnice (pokud kinematická transformace některé nepoužívá, tak mají hodnotu 0), orientaci ve formě eulerovo úhlů ZYX, (jen pokud ji kinematická transformace používá), dodatečné osy (pokud je kinematická transformace používá). Eulerovo úhly a natočení dodatečných rotačních os je v jednotkách definovaných kinematickou transformací (základní jednotka jsou radiány za sekundu).

Vstupy

<code>uAxesGroup</code>	Odkaz na skupinu os	Reference
<code>Enable</code>	Povolení funkce bloku (aktivace výstupů)	Bool
<code>CoordSystem</code>	Volba souřadného systému	Long (I32)
	1 ACM	
	2 MCS	
	3 PCS	

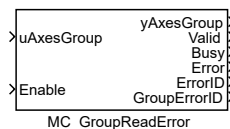
Výstupy

<code>yAxesGroup</code>	Odkaz na skupinu os	Reference
<code>Valid</code>	Příznak platnosti výstupní hodnoty	Bool
<code>Busy</code>	Příznak, že algoritmus ještě neskončil	Bool
<code>Error</code>	Příznak chyby	Bool
<code>ErrorID</code>	Výsledek poslední operace	Error
	i obecná chyba systému REXYGEN	
<code>Velocity</code>	Pole aktuálních rychlostí skupiny os	Reference

MC_GroupReadError – Chyby ve skupině os

Symbol bloku

Licence: [COORDINATED MOTION](#)



Popis funkce

Blok **MC_GroupReadError** zpřístupňuje na výstupu **GroupErrorID** aktuální chybový kód připojené osy. Pokud osa není ve stavu chyby, hodnota tohoto výstupu je 0. Hodnota je platná jen pokud je na výstupu **Valid** true, čehož se dosáhne nastavením vstupu **Enable** na hodnotu true.

Poznámka 1: Tento blok je implementován z důvodu kompatibility s **PLCopen** neboť zobrazuje stejnou veličinu, která je přístupná i na výstupu **ErrorID** bloku [RM_AxesGroup](#).

Vstupy

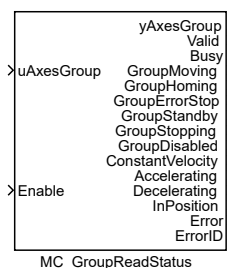
uAxesGroup	Odkaz na skupinu os	Reference
Enable	Povolení funkce bloku (aktivace výstupů)	Bool

Výstupy

yAxesGroup	Odkaz na skupinu os	Reference
Valid	Příznak platnosti výstupní hodnoty	Bool
Busy	Příznak, že algoritmus ještě neskončil	Bool
Error	Příznak chyby	Bool
ErrorID	Výsledek poslední operace i obecná chyba systému REXYGEN	Error
GroupErrorID	Výsledek poslední operace i obecná chyba systému REXYGEN	Error

MC_GroupReadStatus – Stav skupin os

Symbol bloku

Licence: [COORDINATED MOTION](#)

Popis funkce

Blok `MC_GroupReadStatus` indikuje na svých výstupech různé stavy připojené skupiny os jako logickou hodnotu. Indikovaný stav je zřejmý z názvu výstupu, popřípadě z jeho popisu. Hodnota je platná jen pokud je na výstupu `Valid` true, čehož se dosáhne nastavením vstupu `Enable` na hodnotu true.

Vstupy

<code>uAxesGroup</code>	Odkaz na skupinu os	Reference
<code>Enable</code>	Povolení funkce bloku (aktivace výstupů)	Bool

Výstupy

<code>yAxesGroup</code>	Odkaz na skupinu os	Reference
<code>Valid</code>	Příznak platnosti výstupní hodnoty	Bool
<code>Busy</code>	Příznak, že algoritmus ještě neskončil	Bool
<code>GroupMoving</code>	Stav GroupMoving	Bool
<code>GroupHoming</code>	Stav GroupHoming	Bool
<code>GroupErrorStop</code>	Stav ErrorStop	Bool
<code>GroupStandby</code>	Stav Standby	Bool
<code>GroupStopping</code>	Stav Stopping	Bool
<code>GroupDisabled</code>	Stav Disabled	Bool
<code>ConstantVelocity</code>	Pohyb konstantní rychlostí	Bool
<code>Accelerating</code>	Zrychlování	Bool
<code>Decelerating</code>	Zpomalování	Bool
<code>InPosition</code>	Příznak dosažení zadané pozice	Bool
<code>Error</code>	Příznak chyby	Bool

ErrorID	Výsledek poslední operace	Error
i	 obecná chyba systému REXYGEN	

MC_GroupReset – Nulování chyb os ve skupině

Symbol bloku

Licence: [COORDINATED MOTION](#)

Popis funkce

Blok **MC_GroupReset** převede připojenou skupinu os ze stavu „GroupErrorStop“ do stavu „GroupStandBy“ a vynuluje ve skupině všechny příznaky chyby. Blok resetuje také všechny osy v dané skupině os stejně jako blok [MC_Reset](#) z knihovny MC_SINGLE.

Vstupy

uAxesGroup	Odkaz na skupinu os	Reference
Execute	Náběžná hrana aktivuje blok	Bool

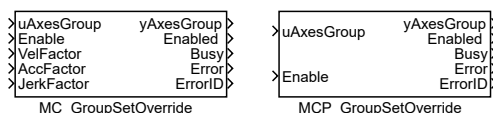
Výstupy

yAxesGroup	Odkaz na skupinu os	Reference
Done	Příznak dokončení algoritmu	Bool
Busy	Příznak, že algoritmus ještě neskončil	Bool
Error	Příznak chyby	Bool
ErrorID	Výsledek poslední operace	Error
	i obecná chyba systému REXYGEN	

MC_GroupSetOverride, MCP_GroupSetOverride – Nastavení násobivých faktorů na osách ve skupině

Symbole bloků

Licence: **COORDINATED MOTION**



Popis funkce

Bloky MC_GroupSetOverride a MCP_GroupSetOverride mají naprosto shodnou funkci, jediným rozdílem je, že MCP_ varianta bloku má méně vstupů a potřebné konstanty se zadávají jako parametry bloku.

Blok MC_GroupSetOverride nastavuje násobivé faktory, které se projeví ve všech blocích pracujících se skupinou os. Hodnoty rychlosti, zrychlení a jerku ve všech blocích je nutné vynásobit faktorem z tohoto bloku, tím dostaneme hodnotu, se kterou blok skutečně pracuje. Toto se netýká limitních hodnot zadáných v **RM_Axis** a administrativních bloků. Tento blok není aktivován hranou, ale pokud je na vstupu **Enable** true, tak se hodnoty trvale aktualizují. Pokud je aktivní například blok typu **MC_MoveLinearAbsolute**, vede to na neustálé přepočítávání trajektorie, což je výpočetně (a tím i časově) náročná operace a navíc se kumulují zaokrouhlovací chyby. Proto je zavedena necitlivost (parametr **diff**) a přepočet trajektorie je proveden, až když se některý z faktorů změní více, než je tato necitlivost.

Poznámka 1: Všechny faktory musí být kladné. Faktory větší než 1 jsou možné, ale často vedou k překročení mezí nastavených na ose a k selhání pohybu (blok hlásí chybu **errorID** = -700 - neplatný parametr) nebo dokonce k havarijnímu zastavení osy (blok pak hlásí chybu **errorID** = -701 - hodnota mimo rozsah).

Poznámka 2: Je dovoleno zadat **VelFactor** = 0. Vede to k dočasnému zastavení aktuálního pohybu (podobně jako **MC_GroupInterrupt**). Pro dokončení pohybu se pak ale musí nastavit **VelFactor** kladný.

Vstupy

		Reference
uAxesGroup	Odkaz na skupinu os	Bool
Enable	Povolení funkce bloku (aktivace výstupů)	Bool
VelFactor	Faktor násobení pro rychlost	Double (F64)
AccFactor	Faktor násobení pro zrychlení	Double (F64)
JerkFactor	Faktor násobení pro změnu zrychlení	Double (F64)

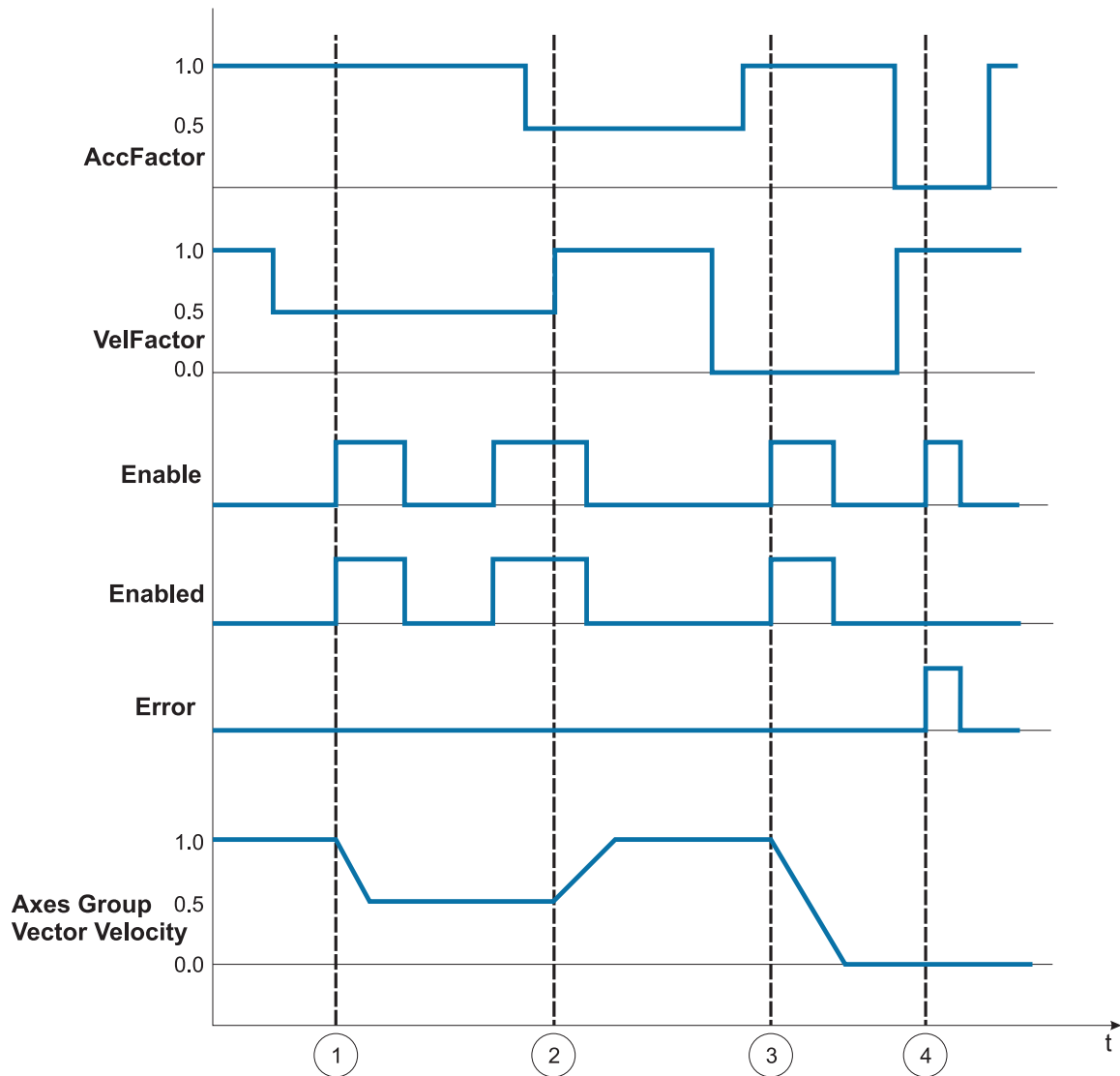
Parametr

diff	Pásmo necitlivosti (pro přepočet trajektorie)	⊙0.05	Double (F64)

Výstupy

		Reference
yAxesGroup	Odkaz na skupinu os	
Enabled	Signalizuje úspěšné nastavení násobivých faktorů	Bool
Busy	Příznak, že algoritmus ještě neskončil	Bool
Error	Příznak chyby	Bool
ErrorID	Výsledek poslední operace	Error
	i obecná chyba systému REXYGEN	

Příklad

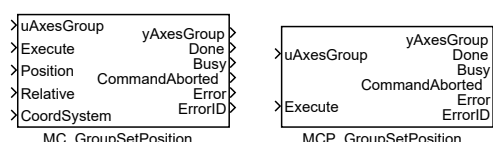


- ① Rychlost skupiny os zmenšena na 50% se 100% zpomalením
- ② Rychlost skupiny os vrácena na 100% s 50% akcelerací
- ③ Rychlost skupiny os zmenšena na 0% se 100% zpomalením
- ④ Žádná změna, protože není povolen $\text{AccFactor} = 0.0$, je sepnuta chyba

MC_GroupSetPosition, MCP_GroupSetPosition – Nastavení polohového offsetu skupiny os

Symbody bloků

Licence: [COORDINATED MOTION](#)



Popis funkce

Bloky MC_GroupSetPosition a MCP_GroupSetPosition mají naprosto shodnou funkci, jediným rozdílem je, že MCP_ varianta bloku má méně vstupů a potřebné konstanty se zadávají jako parametry bloku.

Blok **MC_GroupSetPosition** nastaví polohu všech os ve skupině bez jejich pohybu. Nové souřadnice jsou dány vstupem **Position**. Pomocí vstupu **CoordSystem** se nastaví, v jakém souřadnicovém systému se provede změna. Tato změna následně ovlivní i souřadnice ve vyšších souřadnicových systémech.

POZOR: blok momentálně není implementován, protože není jasný usecase a jak tedy polohy dosáhnout v ACS by mohlo dávat smysl home na osách (už řeší **RM_HomeOffset**), v MCS buď home na osách nebo posunout počátek robota nebo počátek nástroje (tj. řeší **MC_SetCartesianTransform** base offset nebo tool offset), v PCS buď předchozí možnosti nebo změnit transformaci MCS na PCS (řeší **MC_SetCartesianTransform**), v TCS nedává smysl

Vstupy

uAxesGroup	Odkaz na skupinu os	Reference
Execute	Náběžná hrana aktivuje blok	Bool
Position	Pole souřadnic (pozic a orientací)	Reference
Relative	Výběr absolutních (=false) nebo relativních (=true) souřadnic	Bool
	off ... absolute	
	on relative	
CoordSystem	Volba souřadného systému	Long (I32)
	1 ACM	
	2 MCS	
	3 PCS	

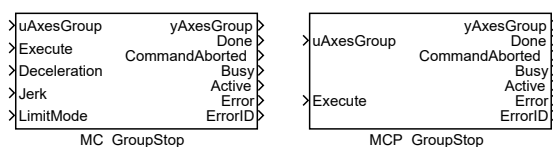
Výstupy

<code>yAxesGroup</code>	Odkaz na skupinu os	Reference
<code>Done</code>	Příznak dokončení algoritmu	Bool
<code>Busy</code>	Příznak, že algoritmus ještě neskončil	Bool
<code>CommandAborted</code>	Příznak přerušení funkce bloku	Bool
<code>Error</code>	Příznak chyby	Bool
<code>ErrorID</code>	Výsledek poslední operace	Error
	<code>i</code> obecná chyba systému REXYGEN	

MC_GroupStop, MCP_GroupStop – Zastavení koordinovaného pohybu

Symbole bloků

Licence: [COORDINATED MOTION](#)



Popis funkce

Bloky MC_GroupStop a MCP_GroupStop mají naprosto shodnou funkci, jediným rozdílem je, že MCP_ varianta bloku má méně vstupů a potřebné konstanty se zadávají jako parametry bloku.

Blok **MC_GroupStop** zastaví pohyb. Režim je vždy aborted, tj. zastavování se spouští okamžitě. Blok se nejprve pokusí zastavit ve směru původní trajektorie. Pokud se to nepodaří, zastaví se pomocí errorstop sekvence na jednotlivých osách již nekoordinovaně. Dokud je na vstup **Execute** hodnota true nebo dokud se skupina pohybuje, nachází se skupina ve stavu „Stopping“ a není možné spouštět další bloky. Okamžitě po zastavení se nastaví na výstupu **Done** hodnota true (pokud nenastane chyba). Skupina os přejde do „Standby“ až po deaktivování vstupu **Execute**.

Poznámka 1: Blok nemá parametr **CoordSystem**, protože jej přejímá z právě běžícího bloku.

Vstupy

uAxesGroup	Odkaz na skupinu os	Reference
Execute	Náběžná hrana aktivuje blok	Bool
Deceleration	Maximální povolené zpomalení [unit/s ²]	Double (F64)
Jerk	Maximální povolená změna zrychlení [unit/s ³]	Double (F64)

Výstupy

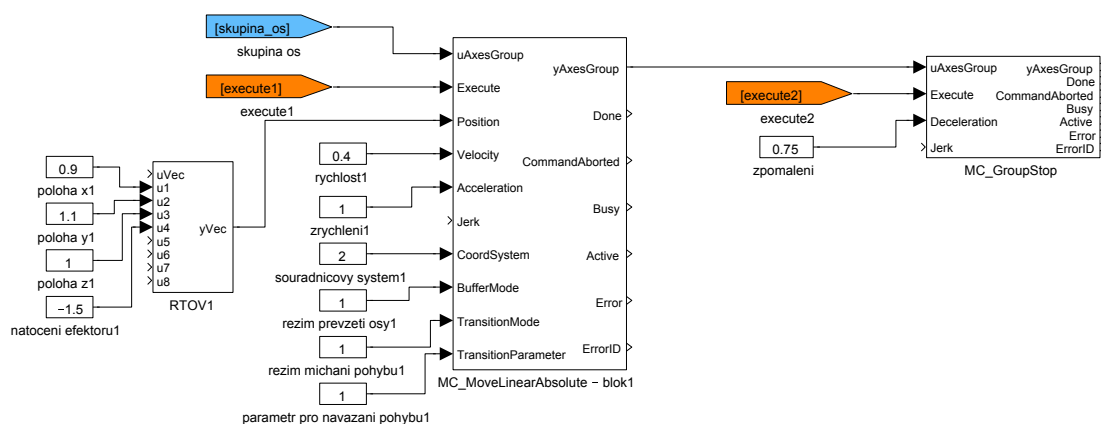
yAxesGroup	Odkaz na skupinu os	Reference
Done	Příznak dokončení algoritmu	Bool
CommandAborted	Příznak přerušení funkce bloku	Bool
Busy	Příznak, že algoritmus ještě neskončil	Bool
Active	Příznak, že blok řídí osu	Bool
Error	Příznak chyby	Bool

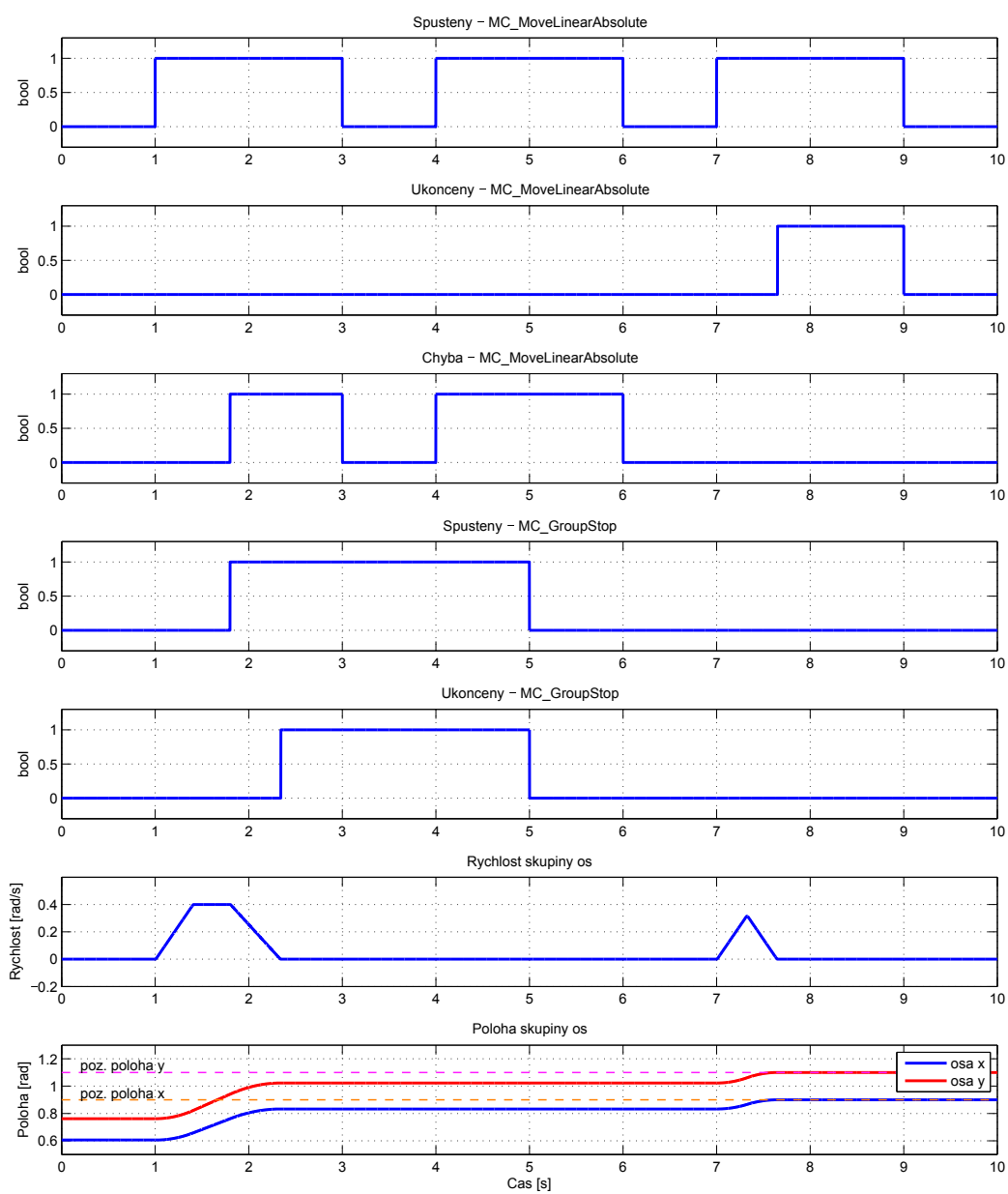
ErrorID Výsledek poslední operace
i obecná chyba systému REXYGEN

Error

Příklad

Na časovém diagramu níže, je uveden příklad na chování bloku MC_GroupStop. Nejprve dojde ke spuštění bloku MC_MoveLinearAbsolute a tím dojde k uvedení osy do pohybu. Následně je již spuštěn blok MC_GroupStop, který způsobí zastavení skupiny os. Následně druhé spuštění exekutivy bloku MC_MoveLinearAbsolute nic nezpůsobí a to i přes to, že je skupina již v klidu. To z toho důvodu, že je stále spuštěný blok MC_GroupStop. Až třetí spuštění exekutivy bloku MC_MoveLinearAbsolute dostane skupinu os do požadované polohy.

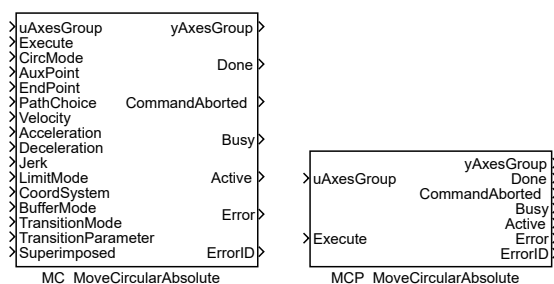




MC_MoveCircularAbsolute, MCP_MoveCircularAbsolute – Pohyb do pozice po kružnicích (absolutní souřadnice)

Symbyoly bloků

Licence: [COORDINATED MOTION](#)



Popis funkce

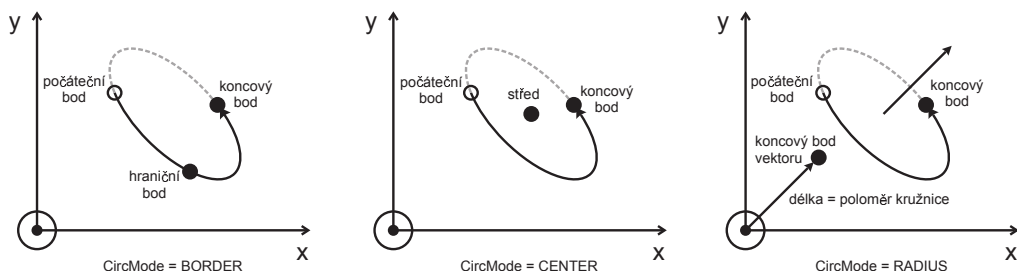
Bloky MC_MoveCircularAbsolute a MCP_MoveCircularAbsolute mají naprosto shodnou funkci, jediným rozdílem je, že MCP_ varianta bloku má méně vstupů a potřebné konstanty se zadávají jako parametry bloku.

Blok **MC_MoveCircularAbsolute** slouží pro přesun koncového efektoru po kružnici (resp. kruhovém oblouku) na zadanou pozici. Pozice se zadává absolutně v souřadném systému zvoleném vstupem **CoordSystem**. Parametry **Velocity**, **Acceleration**, **Deceleration** a **Jerk** určují rychlost, zrychlení, zpomalení a změnu zrychlení ve směru pohybu (tj. tečně k trajektorii). Pro určení těchto parametrů se vychází při použití souřadného systému MCS nebo PCS jen z polohových souřadnic. Další souřadnice (úhel natočení) se již generují proporcionálně, tak aby byl pohyb v těchto souřadnicích lineární a skončil ve stejném okamžiku jako polohové souřadnice. Pokud se poloha nemění (dochází tedy jen k otočení koncového efektoru), počítá se rychlost/zrychlení ze všech souřadnic, ale číslo má pak jiný fyzikální význam. Cílovou polohu určuje vektorový parametr **EndPoint**. Pomocný bod je určen vektorovým parametrem **AuxPoint**. Tyto parametry musí mít tolik prvků, kolik předpokládá kinematická transformace (viz [MC_SetKinTransform_Lin](#)). V opačném případě je signalizována chyba a pohyb se neprovede. Význam pomocného bodu je uveden níže. Pohyb je spuštěn náběžnou hranou na vstupu **Execute**. K dispozici jsou následující způsoby zadání kružnice:

BORDER – Zadává se koncový bod a bod, kterým má kružnice procházet (vstup **AuxPoint**).

CENTER – Zadává se koncový bod a střed kružnice (vstup **AuxPoint**). Vstup **PathChoice** potom definuje, zda bude vygenerovaný pohyb po směru nebo proti směru hodinových ručiček.

RADIUS – Zadává se koncový bod a vektor kolmý k rovině ve které se má nacházet kružnice. Délka vektoru udává poloměr kružnice. Příklad: Vstup **AuxPoint** = (50,0,0) odpovídá kružnici v rovině y-z s poloměrem 50 a rotaci kolem osy paralelní s osou x odpovídající pravidlu pravé ruky (**CoordSystem** = MCS).



Poznámka 1: v režimu „RADIUS“ nemá pomocný bod význam polohy a proto se zadává vždy absolutně.

Poznámka 2: Každý ze způsobu zadávání (parametr **CircMode**) má některé výhody a nevýhody. Žádný z uvedených způsobů neumožňuje „projet“ celý kruh nebo dokonce několik „otáček“. Dále vzniká problém v případě půlkruhu, kdy v režimu zadání středu není definována rovina kružnice. Při zadání středu je kružnice přerušena a je nutné dát pozor, aby vzdálenost počátečního a koncového bodu byla stejná (v opačném případě blok skončí s chybou a trajektorie se negeneruje).

Poznámka 3: U kružnic v třírozměrném prostoru je problém definovat kladný směr. Algoritmus používá metodu vektorového součinu počátečního a koncového průvodiče. Tím dostaneme normálový vektor, který určuje „směr nahoru“ a pak již je kladný směr zřejmý. Metoda ovšem nefunguje pro půlkružnici a také je potřeba dát pozor při posunutí koncového nebo počátečního bodu, protože může dojít ke změně orientace (kladný směr se stane záporným).

Poznámka 4: Podle typu předchozího pohybu a v některých případech i podle parametrů (zejména v případě příliš krátké trajektorie) nemusí být implementované nebo realizovatelné všechny varianty **TransitionMode**.

Vstupy

uAxesGroup	Odkaz na skupinu os	Reference
Execute	Náběžná hrana aktivuje blok	Bool
CircMode	Určuje význam vstupních signálů AuxPoint a CircDirection	Long (I32)
	1 BORDER	
	2 CENTER	
	3 RADIUS	
AuxPoint	Absolutně zadaná pozice	Reference
EndPoint	Absolutně zadaná pozice koncového bodu	Reference
PathChoice	Volba směru	Long (I32)
	1 Ve směru hodinových ručiček	
	2 Protisměru hodinových ručiček	
Velocity	Maximální povolená rychlost [unit/s]	Double (F64)
Acceleration	Maximální povolené zrychlení [unit/s ²]	Double (F64)

Jerk	Maximální povolená změna zrychlení [unit/s ³]	Double (F64)
LimitMode	Volba jednotek pro limity (Velocity, Acceleration, Jerk) ☉1	Long (I32)
	1 Relative [part of group limit]	
	2 Absolute [unit/s, unit/s ² , unit/s ³]	
CoordSystem	Volba souřadného systému	Long (I32)
	1 ACM	
	2 MCS	
	3 PCS	
BufferMode	Režim převzetí osy	Long (I32)
	1 Aborting (nový blok se spustí okamžitě)	
	2 Buffered (nový blok se spustí po dokončení předchozího)	
	3 Blending low (nový blok se spustí po dokončení předchozího, původní pohyb skončí s nižší rychlostí z obou bloků)	
	4 Blending high (nový blok se spustí po dokončení předchozího, původní pohyb skončí s vyšší rychlostí z obou bloků)	
	5 Blending previous (nový blok se spustí po dokončení předchozího, původní pohyb skončí se svojí koncovou rychlostí)	
	6 Blending next (nový blok se spustí po dokončení předchozího, původní pohyb skončí s počáteční rychlostí nového bloku)	
TransitionMode	Režim míchání pohybu	Long (I32)
	1 TMNone (xx)	
	2 TMstartvelocity (proložení s danou počáteční rychlostí)	
	3 TMConstantVelocity (proložení s danou konstantní rychlostí)	
	4 TMCornerDistance (xx)	
	5 TMMaxCornerDeviation (xx)	
	11 Smooth (nový blok se spustí po dokončení předchozího, původní pohyb skončí s počáteční rychlostí nového bloku)	
TransitionParameter	Parametr pro navázání pohybu (dle zvoleného režimu míchání)	Double (F64)
Superimposed	Příznak vykonání jako vedlejší (superimposed) pohyb	Bool

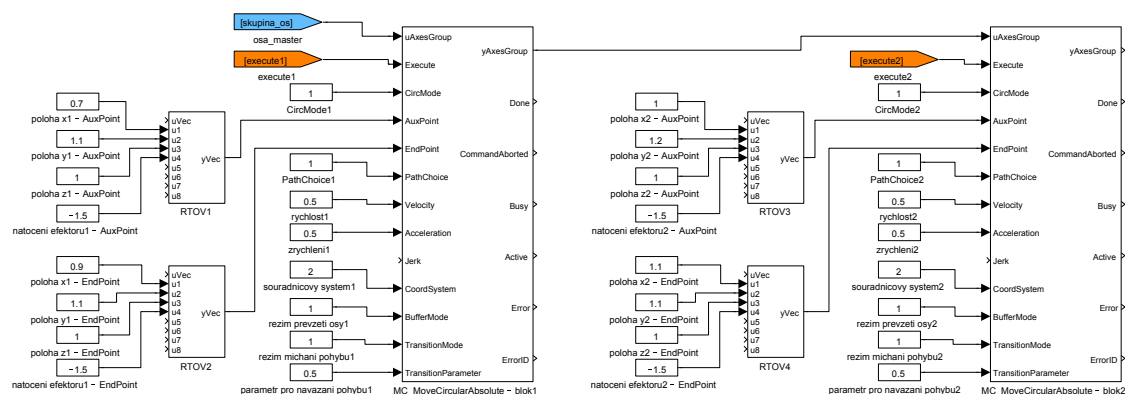
Výstupy

yAxesGroup	Odkaz na skupinu os	Reference
Done	Příznak dokončení algoritmu	Bool
CommandAborted	Příznak přerušení funkce bloku	Bool
Busy	Příznak, že algoritmus ještě neskončil	Bool
Active	Příznak, že blok řídí osu	Bool
Error	Příznak chyby	Bool

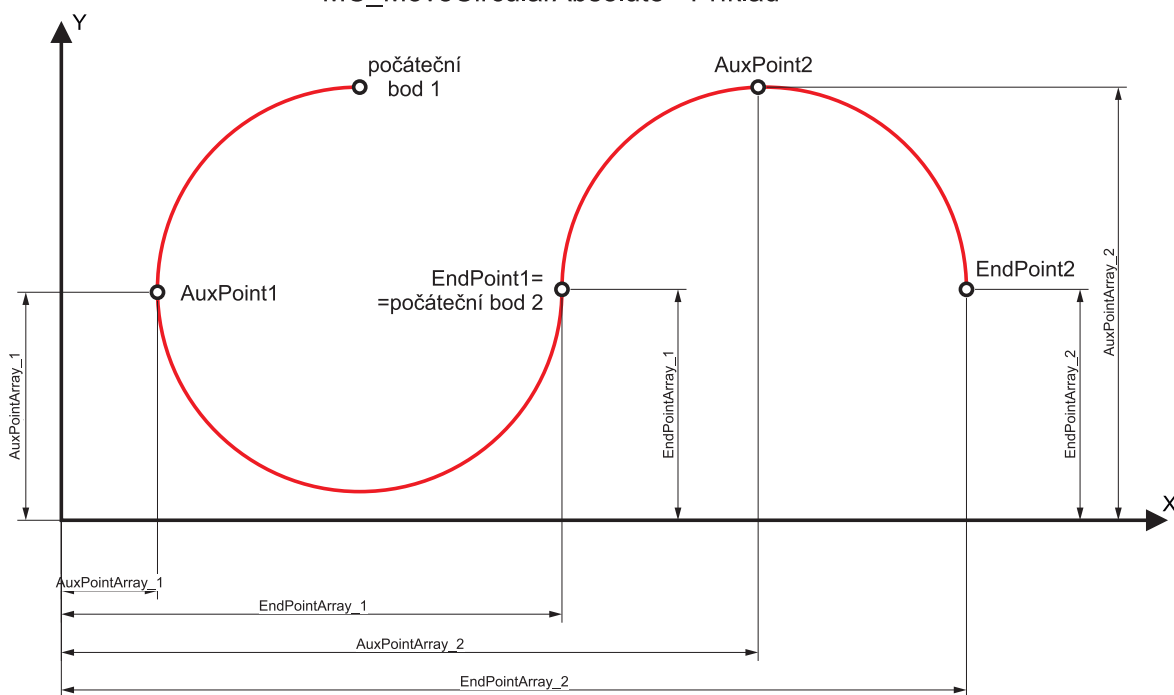
ErrorID Výsledek poslední operace
i obecná chyba systému REXYGEN

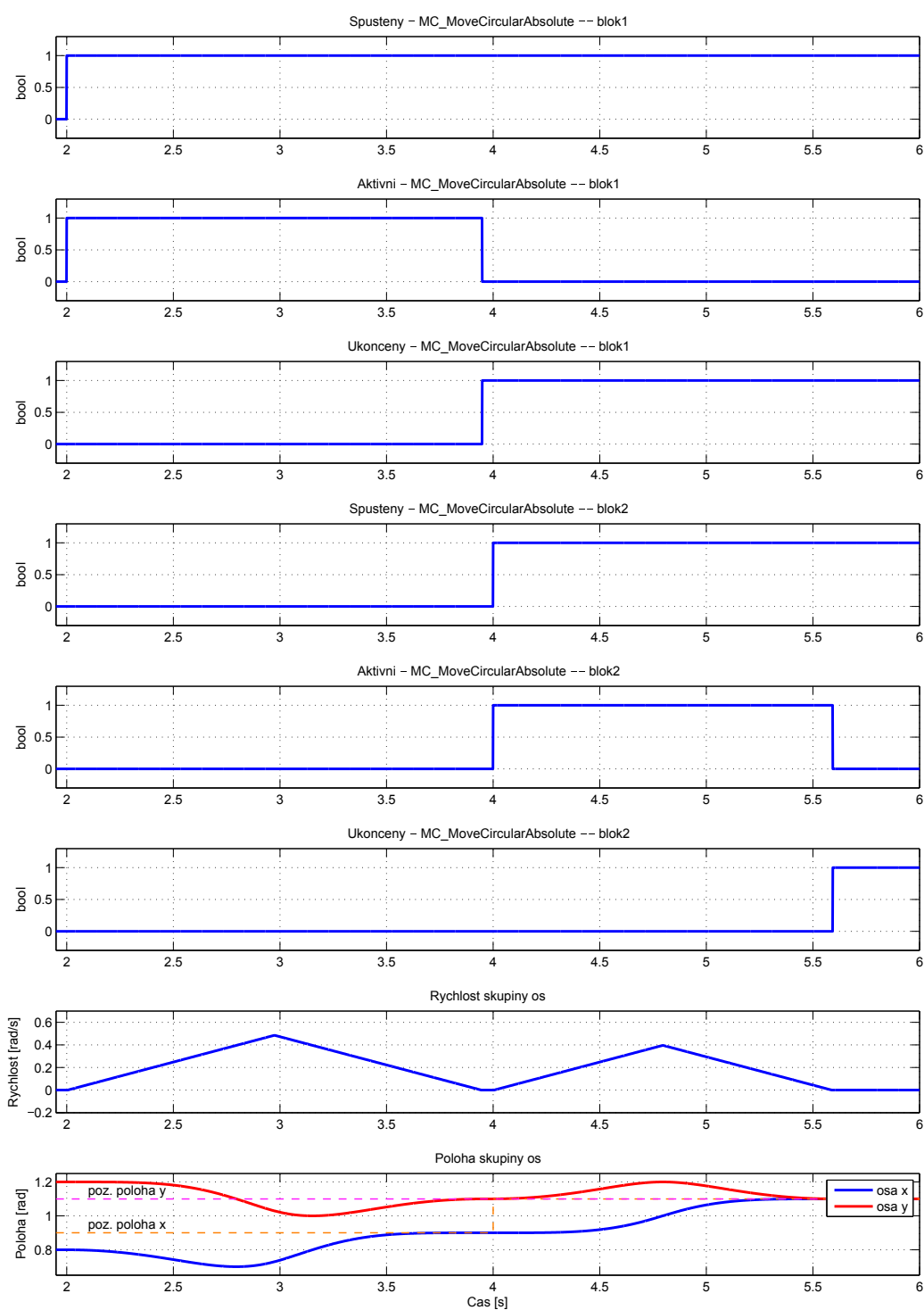
Error

Příklad



MC_MoveCircularAbsolute - Příklad

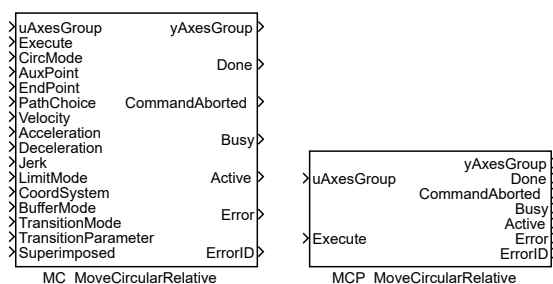




MC_MoveCircularRelative, MCP_MoveCircularRelative – Pohyb do pozice po kružnicích (relativní souřadnice)

Symboly bloků

Licence: [COORDINATED MOTION](#)



Popis funkce

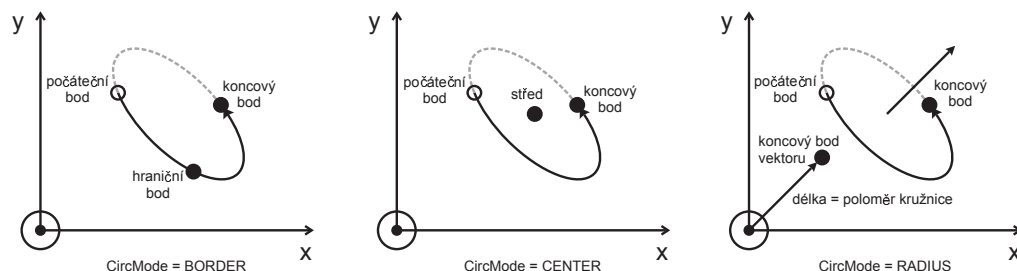
Bloky MC_MoveCircularRelative a MCP_MoveCircularRelative mají naprosto shodnou funkci, jediným rozdílem je, že MCP_ varianta bloku má méně vstupů a potřebné konstanty se zadávají jako parametry bloku.

Blok **MC_MoveCircularRelative** slouží pro přesun koncového efektoru po kružnici (resp. kruhovém oblouku) na zadanou pozici. Pozice se zadává relativně od aktuální polohy v souřadném systému zvoleném vstupem **CoordSystem**. Parametry **Velocity**, **Acceleration**, **Deceleration** a **Jerk** určují rychlost, zrychlení, zpomalení a změnu zrychlení ve směru pohybu (tj. tečně k trajektorii). Pro určení těchto parametrů se vychází při použití souřadného systému MCS nebo PCS jen z polohových souřadnic. Další souřadnice (úhel natočení) se již generují proporcionálně, tak aby byl pohyb v těchto souřadnicích lineární a skončil ve stejném okamžiku jako polohové souřadnice. Pokud se poloha nemění (dochází tedy jen k otočení koncového efektoru), počítá se rychlost/zrychlení ze všech souřadnic, ale číslo má pak jiný fyzikální význam. Cílovou polohu určuje vektorový parametr **EndPoint**. Pomocný bod je určen vektorovým parametrem **AuxPoint**. Tyto parametry musí mít tolik prvků, kolik předpokládá kinematická transformace (viz [MC_SetKinTransform_Lin](#)). V opačném případě je signalizována chyba a pohyb se neprovede. Význam pomocného bodu je uveden níže. Pohyb je spuštěn náběžnou hranou na vstupu **Execute**. K dispozici jsou následující způsoby zadání kružnice:

BORDER – Zadává se koncový bod a bod, kterým má kružnice procházet (vstup **AuxPoint**).

CENTER – Zadává se koncový bod a střed kružnice (vstup **AuxPoint**). Vstup **PathChoice** potom definuje, zda bude vygenerovaný pohyb po směru nebo proti směru hodinových ručiček.

RADIUS – Zadává se koncový bod a vektor kolmý k rovině ve které se má nacházet kružnice. Délka vektoru udává poloměr kružnice. Příklad: Vstup **AuxPoint** = (50,0,0) odpovídá kružnici v rovině y-z s poloměrem 50 a rotaci kolem osy paralelní s osou x odpovídající pravidlu pravé ruky (**CoordSystem** = MCS).



Poznámka 1: v režimu „RADIUS“ nemá pomocný bod význam polohy a proto se zadává vždy absolutně.

Poznámka 2: Každý ze způsobu zadávání (parametr **CircMode**) má některé výhody a nevýhody. Žádný z uvedených způsobů neumožňuje „projet“ celý kruh nebo dokonce několik „otáček“. Dále vzniká problém v případě půlkruhu, kdy v režimu zadání středu není definována rovina kružnice. Při zadání středu je kružnice přeurlčená a je nutné dát pozor, aby vzdálenost počátečního a koncového bodu byla stejná (v opačném případě blok skončí s chybou a trajektorie se negeneruje).

Poznámka 3: U kružnic v třírozměrném prostoru je problém definovat kladný směr. Algoritmus používá metodu vektorového součinu počátečního a koncového průvodiče. Tím dostaneme normálový vektor, který určuje „směr nahoru“ a pak již je kladný směr zřejmý. Metoda ovšem nefunguje pro půlkružnici a také je potřeba dát pozor při posunutí koncového nebo počátečního bodu, protože může dojít ke změně orientace (kladný směr se stane záporným).

Poznámka 4: Podle typu předchozího pohybu a v některých případech i podle parametrů (zejména v případě příliš krátké trajektorie) nemusí být implementované nebo realizovatelné všechny varianty **TransitionMode**.

Vstupy

uAxesGroup	Odkaz na skupinu os	Reference
Execute	Náběžná hrana aktivuje blok	Bool
CircMode	Určuje význam vstupních signálů AuxPoint a CircDirection	Long (I32)
	1 BORDER	
	2 CENTER	
	3 RADIUS	
AuxPoint	Absolutně zadaná pozice	Reference
EndPoint	Absolutně zadaná pozice koncového bodu	Reference
PathChoice	Volba směru	Long (I32)
	1 Ve směru hodinových ručiček	
	2 Protisměru hodinových ručiček	
Velocity	Maximální povolená rychlost [unit/s]	Double (F64)
Acceleration	Maximální povolené zrychlení [unit/s ²]	Double (F64)

Jerk	Maximální povolená změna zrychlení [unit/s ³]	Double (F64)
LimitMode	Volba jednotek pro limity (Velocity, Acceleration, Jerk) ☉1	Long (I32)
	1 Relative [part of group limit]	
	2 Absolute [unit/s, unit/s ² , unit/s ³]	
CoordSystem	Volba souřadného systému	Long (I32)
	1 ACM	
	2 MCS	
	3 PCS	
BufferMode	Režim převzetí osy	Long (I32)
	1 Aborting (nový blok se spustí okamžitě)	
	2 Buffered (nový blok se spustí po dokončení předchozího)	
	3 Blending low (nový blok se spustí po dokončení předchozího, původní pohyb skončí s nižší rychlostí z obou bloků)	
	4 Blending high (nový blok se spustí po dokončení předchozího, původní pohyb skončí s vyšší rychlostí z obou bloků)	
	5 Blending previous (nový blok se spustí po dokončení předchozího, původní pohyb skončí se svojí koncovou rychlostí)	
	6 Blending next (nový blok se spustí po dokončení předchozího, původní pohyb skončí s počáteční rychlostí nového bloku)	
TransitionMode	Režim míchání pohybu	Long (I32)
	1 TMNone (xx)	
	2 TMstartvelocity (proložení s danou počáteční rychlostí)	
	3 TMConstantVelocity (proložení s danou konstantní rychlostí)	
	4 TMCornerDistance (xx)	
	5 TMMaxCornerDeviation (xx)	
	11 Smooth (nový blok se spustí po dokončení předchozího, původní pohyb skončí s počáteční rychlostí nového bloku)	
TransitionParameter	Parametr pro navázání pohybu (dle zvoleného režimu míchání)	Double (F64)
Superimposed	Příznak vykonání jako vedlejší (superimposed) pohyb	Bool

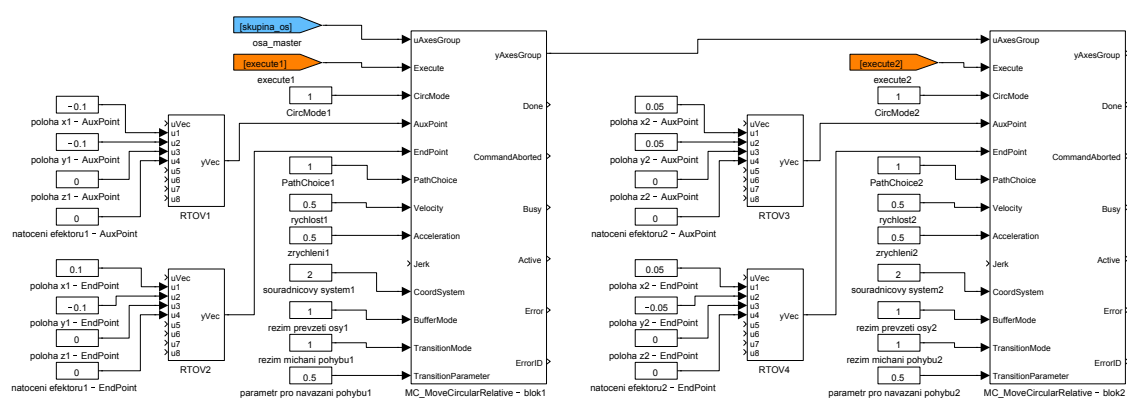
Výstupy

yAxesGroup	Odkaz na skupinu os	Reference
Done	Příznak dokončení algoritmu	Bool
CommandAborted	Příznak přerušení funkce bloku	Bool
Busy	Příznak, že algoritmus ještě neskončil	Bool
Active	Příznak, že blok řídí osu	Bool
Error	Příznak chyby	Bool

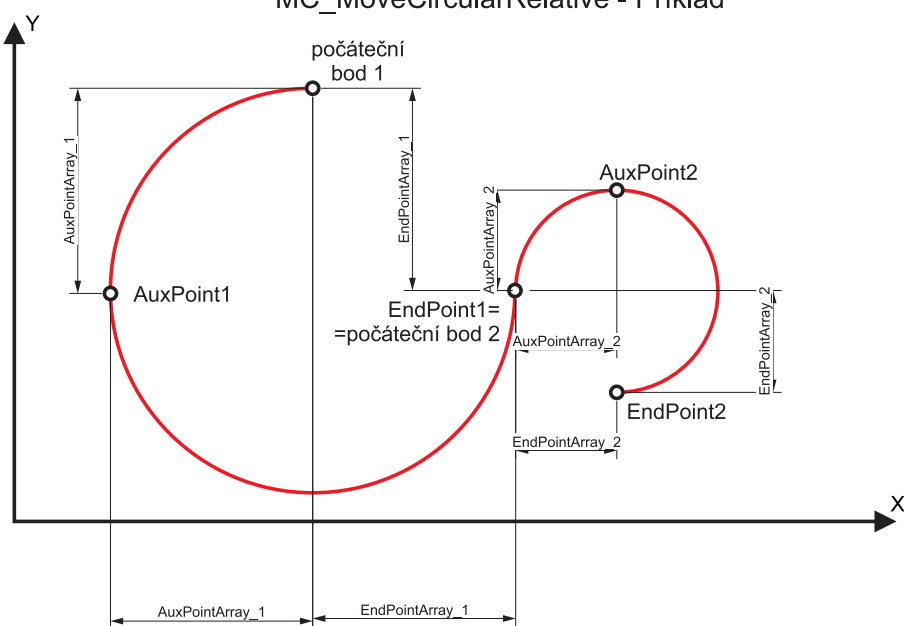
ErrorID Výsledek poslední operace
i obecná chyba systému REXYGEN

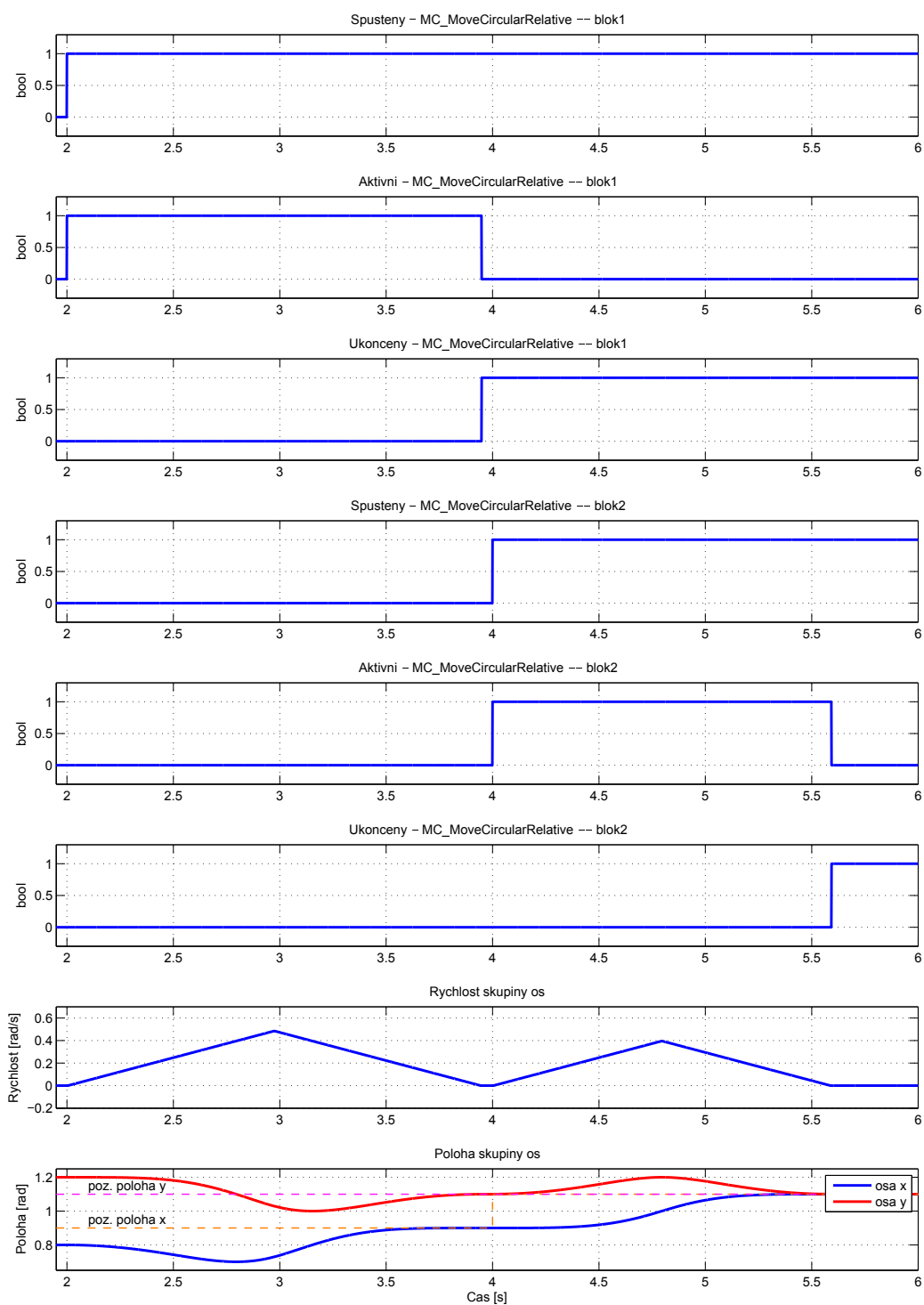
Error

Příklad



MC_MoveCircularRelative - Příklad

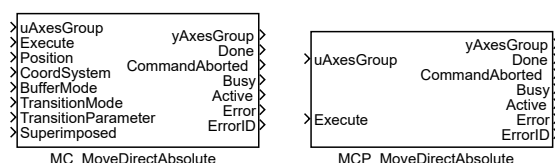




MC_MoveDirectAbsolute, MCP_MoveDirectAbsolute – Nekoordinovaný pohyb do pozice (absolutní souřadnice)

Symbole bloků

Licence: [COORDINATED MOTION](#)



Popis funkce

Bloky MC_MoveDirectAbsolute a MCP_MoveDirectAbsolute mají naprosto shodnou funkci, jediným rozdílem je, že MCP_ varianta bloku má méně vstupů a potřebné konstanty se zadávají jako parametry bloku.

Blok **MC_MoveDirectAbsolute** slouží pro co nejrychlejší přesun stroje (skupiny os) na zadanou pozici, přičemž nezáleží na přesné trajektorii. Tento blok proto nemá parametry určující rychlost a zrychlení. Pohybuje se s maximální rychlostí a zrychlením nastavenými na jednotlivých osách/motorech. Trajektorie je generována tak, aby všechny motory dokončili pohyb ve stejnou dobu, proto se některé motory mohou pohybovat pomaleji, než je jejich limit. Pozice se zadává v absolutně v souřadném systému zvoleném vstupem **CoordSystem**. Tento parametr musí mít tolik prvků, kolik předpokládá kinematická transformace (viz [MC_SetKinTransform_Lin](#)). V opačném případě je signalizována chyba a pohyb se neprovede. Pohyb je spuštěn náběžnou hranou na vstupu **Execute**.

Poznámka: Podle typu předchozího pohybu a v některých případech i podle parametrů (zejména v případě příliš krátké trajektorie) nemusí být implementované nebo realizovatelné všechny varianty **TransitionMode**.

Vstupy

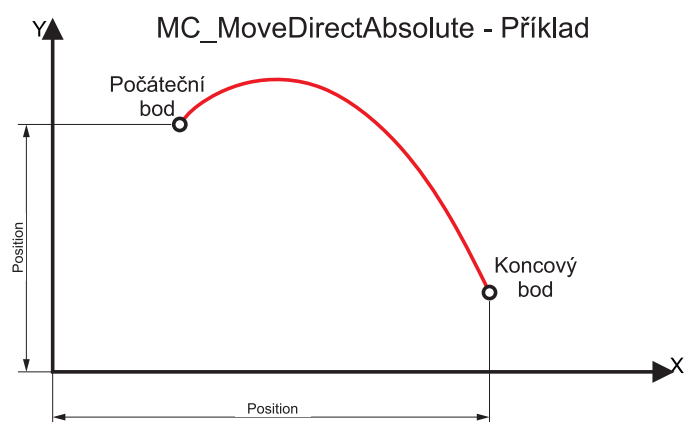
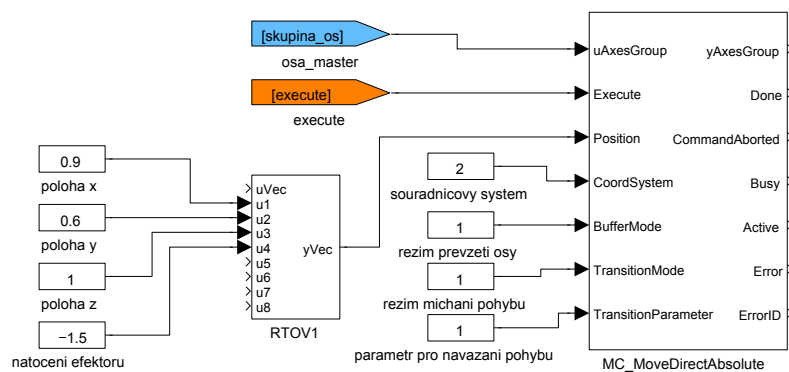
uAxesGroup	Odkaz na skupinu os	Reference
Execute	Náběžná hrana aktivuje blok	Bool
Position	Pole souřadnic (pozic a orientací)	Reference
CoordSystem	Volba souřadného systému	Long (I32)
	1 ACM	
	2 MCS	
	3 PCS	

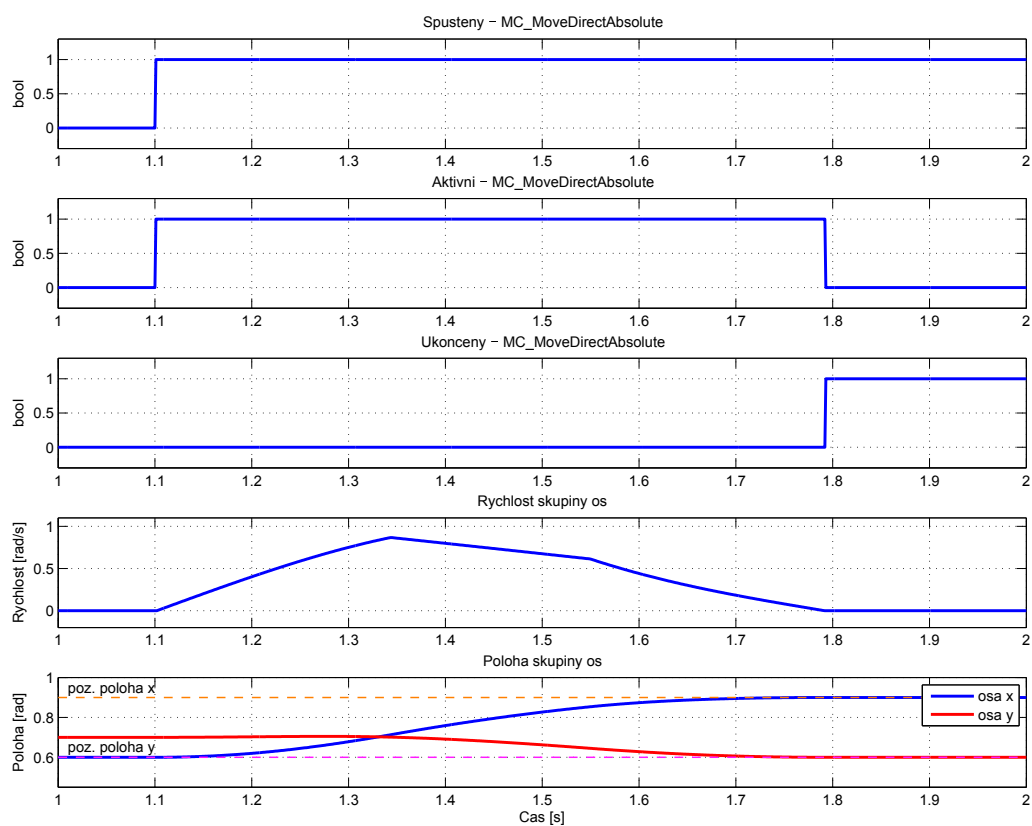
BufferMode	Režim převzetí osy	Long (I32)
1	 Aborting (nový blok se spustí okamžitě)	
2	 Buffered (nový blok se spustí po dokončení předchozího)	
3	 Blending low (nový blok se spustí po dokončení předchozího, původní pohyb skončí s nižší rychlostí z obou bloků)	
4	 Blending high (nový blok se spustí po dokončení předchozího, původní pohyb skončí s vyšší rychlostí z obou bloků)	
5	 Blending previous (nový blok se spustí po dokončení předchozího, původní pohyb skončí se svojí koncovou rychlostí)	
6	 Blending next (nový blok se spustí po dokončení předchozího, původní pohyb skončí s počáteční rychlostí nového bloku)	
TransitionMode	Režim míchání pohybu	Long (I32)
1	 TMNone (xx)	
2	 TMstartvelocity (proložení s danou počáteční rychlostí)	
3	 TMConstantVelocity (proložení s danou konstantní rychlostí)	
4	 TMCornerDistance (xx)	
5	 TMMaxCornerDeviation (xx)	
11	 Smooth (nový blok se spustí po dokončení předchozího, původní pohyb skončí s počáteční rychlostí nového bloku)	
TransitionParameter	Parametr pro navázání pohybu (dle zvoleného režimu míchání)	Double (F64)
Superimposed	Příznak vykonání jako vedlejší (superimposed) pohyb	Bool

Výstupy

yAxesGroup	Odkaz na skupinu os	Reference
Done	Příznak dokončení algoritmu	Bool
CommandAborted	Příznak přerušení funkce bloku	Bool
Busy	Příznak, že algoritmus ještě neskončil	Bool
Active	Příznak, že blok řídí osu	Bool
Error	Příznak chyby	Bool
ErrorID	Výsledek poslední operace	Error
i	 obecná chyba systému REXYGEN	

Příklad

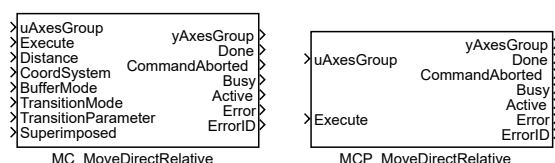




MC_MoveDirectRelative, MCP_MoveDirectRelative – Nekoordinovaný pohyb do pozice (relativní souřadnice)

Symbody bloků

Licence: [COORDINATED MOTION](#)



Popis funkce

Bloky MC_MoveDirectRelative a MCP_MoveDirectRelative mají naprosto shodnou funkci, jediným rozdílem je, že MCP_ varianta bloku má méně vstupů a potřebné konstanty se zadávají jako parametry bloku.

Blok **MC_MoveDirectAbsolute** slouží pro co nejrychlejší přesun stroje (skupiny os) na zadanou pozici, přičemž nezáleží na přesné trajektorii. Tento blok proto nemá parametry určující rychlost a zrychlení. Pohybuje se s maximální rychlostí a zrychlením nastavenými na jednotlivých osách/motorech. Trajektorie je generována tak, aby všechny motory dokončili pohyb ve stejnou dobu, proto se některé motory mohou pohybovat pomaleji, než je jejich limit. Pozice se zadává relativně od aktuální polohy v souřadném systému zvoleném vstupem **CoordSystem**. Tento parametr musí mít tolik prvků, kolik předpokládá kinematická transformace (viz [MC_SetKinTransform_Lin](#)). V opačném případě je signalizována chyba a pohyb se neprovede. Pohyb je spuštěn náběžnou hranou na vstupu **Execute**.

Poznámka: Podle typu předchozího pohybu a v některých případech i podle parametrů (zejména v případě příliš krátké trajektorie) nemusí být implementované nebo realizovatelné všechny varianty **TransitionMode**.

Vstupy

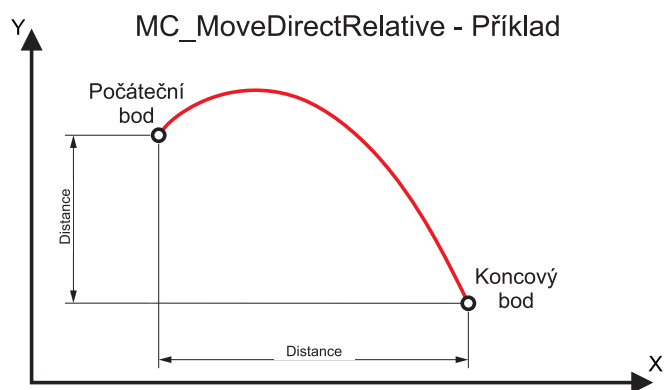
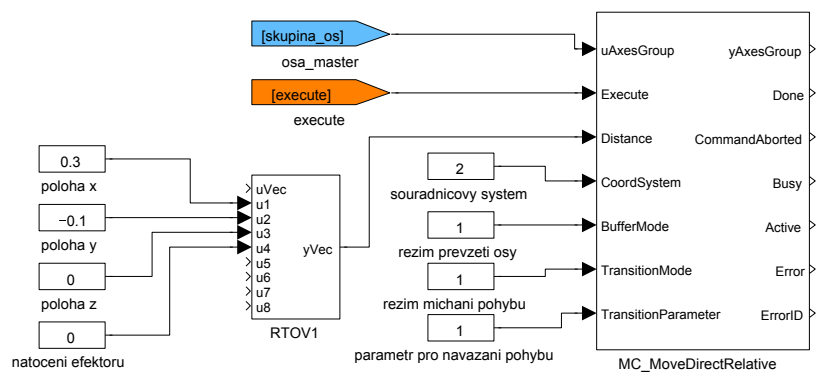
uAxesGroup	Odkaz na skupinu os	Reference
Execute	Náběžná hrana aktivuje blok	Bool
Distance	Pole souřadnic (relativních pozic a orientací)	Reference
CoordSystem	Volba souřadného systému	Long (I32)
	1 ACM	
	2 MCS	
	3 PCS	

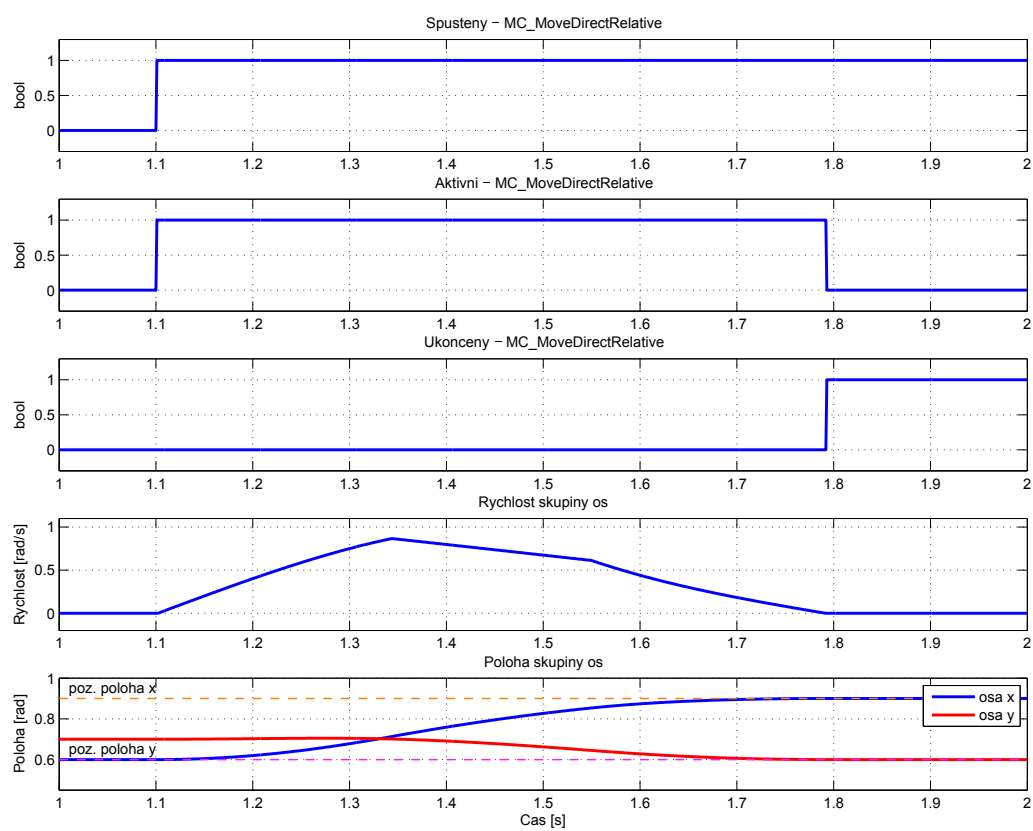
BufferMode	Režim převzetí osy	Long (I32)
1	 Aborting (nový blok se spustí okamžitě)	
2	 Buffered (nový blok se spustí po dokončení předchozího)	
3	 Blending low (nový blok se spustí po dokončení předchozího, původní pohyb skončí s nižší rychlostí z obou bloků)	
4	 Blending high (nový blok se spustí po dokončení předchozího, původní pohyb skončí s vyšší rychlostí z obou bloků)	
5	 Blending previous (nový blok se spustí po dokončení předchozího, původní pohyb skončí se svojí koncovou rychlostí)	
6	 Blending next (nový blok se spustí po dokončení předchozího, původní pohyb skončí s počáteční rychlostí nového bloku)	
TransitionMode	Režim míchání pohybu	Long (I32)
1	 TMNone (xx)	
2	 TMstartvelocity (proložení s danou počáteční rychlostí)	
3	 TMConstantVelocity (proložení s danou konstantní rychlostí)	
4	 TMCornerDistance (xx)	
5	 TMMaxCornerDeviation (xx)	
11	 Smooth (nový blok se spustí po dokončení předchozího, původní pohyb skončí s počáteční rychlostí nového bloku)	
TransitionParameter	Parametr pro navázání pohybu (dle zvoleného režimu míchání)	Double (F64)
Superimposed	Příznak vykonání jako vedlejší (superimposed) pohyb	Bool

Výstupy

yAxesGroup	Odkaz na skupinu os	Reference
Done	Příznak dokončení algoritmu	Bool
CommandAborted	Příznak přerušení funkce bloku	Bool
Busy	Příznak, že algoritmus ještě neskončil	Bool
Active	Příznak, že blok řídí osu	Bool
Error	Příznak chyby	Bool
ErrorID	Výsledek poslední operace	Error
i	 obecná chyba systému REXYGEN	

Příklad

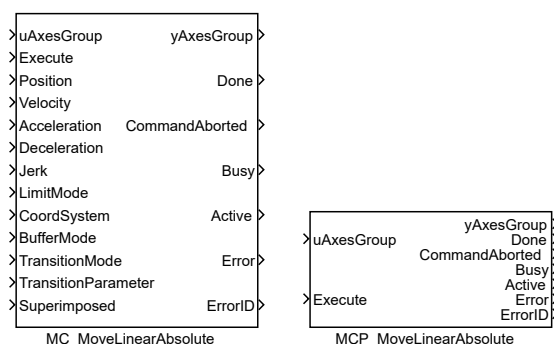




MC_MoveLinearAbsolute, MCP_MoveLinearAbsolute – Pohyb do pozice po přímkách (absolutní souřadnice)

Symbody bloků

Licence: [COORDINATED MOTION](#)



Popis funkce

Bloky MC_MoveLinearAbsolute a MCP_MoveLinearAbsolute mají naprosto shodnou funkci, jediným rozdílem je, že MCP_ varianta bloku má méně vstupů a potřebné konstanty se zadávají jako parametry bloku.

Blok **MC_MoveLinearAbsolute** slouží pro přesun koncového efektoru po přímce na zadanou pozici. Pozice se zadává absolutně v souřadném systému zvoleném vstupem **CoordSystem**. Parametry **Velocity**, **Acceleration**, **Deceleration** a **Jerk** určují rychlost, zrychlení, zpomalení a změnu zrychlení ve směru pohybu (tj. tečně k trajektorii). Pro určení těchto parametrů se vychází při použití souřadného systému MCS nebo PCS jen z polohových souřadnic. Další souřadnice (úhel natočení) se již generují proporcionálně, tak aby byl pohyb v těchto souřadnicích lineární a skončil ve stejném okamžiku jako polohové souřadnice. Pokud se poloha nemění (dochází tedy jen k otočení koncového efektoru), počítá se rychlost/zrychlení ze všech souřadnic, ale číslo má pak jiný fyzikální význam. Cílovou polohu určuje vektorový parametr **Position**. Tento parametr musí mít tolik prvků, kolik předpokládá kinematická transformace (viz [MC_SetKinTransform_Lin](#)). V opačném případě je signalizována chyba a pohyb se neprovede. Pohyb je spuštěn náběžnou hranou na vstupu **Execute**.

Poznámka: Podle typu předchozího pohybu a v některých případech i podle parametrů (zejména v případě příliš krátké trajektorie) nemusí být implementované nebo realizovatelné všechny varianty **TransitionMode**.

Vstupy

uAxesGroup Odkaz na skupinu os

Reference

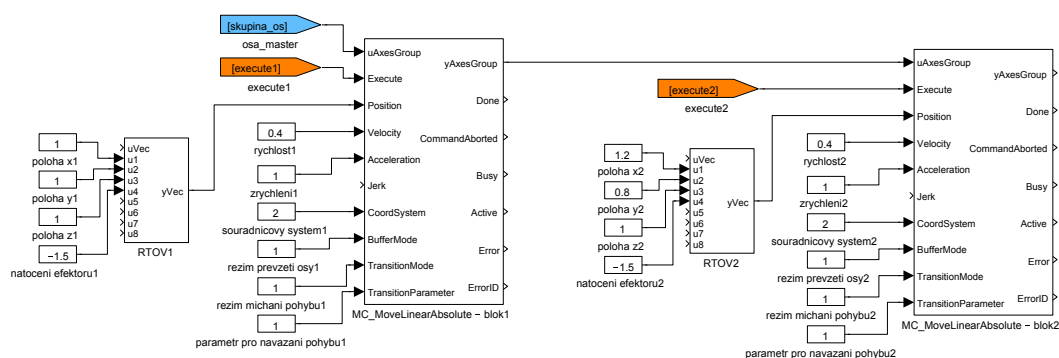
Execute	Náběžná hrana aktivuje blok	Bool
Position	Pole souřadnic (pozic a orientací)	Reference
Velocity	Maximální povolená rychlost [unit/s]	Double (F64)
Acceleration	Maximální povolené zrychlení [unit/s ²]	Double (F64)
Jerk	Maximální povolená změna zrychlení [unit/s ³]	Double (F64)
LimitMode	Volba jednotek pro limity (Velocity, Acceleration, Jerk) 1 Relative [part of group limit] 2 Absolute [unit/s, unit/s ² , unit/s ³]	⊙1 Long (I32)
CoordSystem	Volba souřadného systému 1 ACM 2 MCS 3 PCS	Long (I32)
BufferMode	Režim převzetí osy 1 Aborting (nový blok se spustí okamžitě) 2 Buffered (nový blok se spustí po dokončení předchozího) 3 Blending low (nový blok se spustí po dokončení předchozího, původní pohyb skončí s nižší rychlostí z obou bloků) 4 Blending high (nový blok se spustí po dokončení předchozího, původní pohyb skončí s vyšší rychlostí z obou bloků) 5 Blending previous (nový blok se spustí po dokončení předchozího, původní pohyb skončí se svojí koncovou rychlostí) 6 Blending next (nový blok se spustí po dokončení předchozího, původní pohyb skončí s počáteční rychlostí nového bloku)	Long (I32)
TransitionMode	Režim míchání pohybu 1 TMNone (xx) 2 TMstartvelocity (proložení s danou počáteční rychlostí) 3 TMConstantVelocity (proložení s danou konstantní rychlostí) 4 TMCornerDistance (xx) 5 TMMaxCornerDeviation (xx) 11 Smooth (nový blok se spustí po dokončení předchozího, původní pohyb skončí s počáteční rychlostí nového bloku)	Long (I32)
TransitionParameter	Parametr pro navázání pohybu (dle zvoleného režimu míchání)	Double (F64)
Superimposed	Příznak vykonání jako vedlejší (superimposed) pohyb	Bool

Výstupy

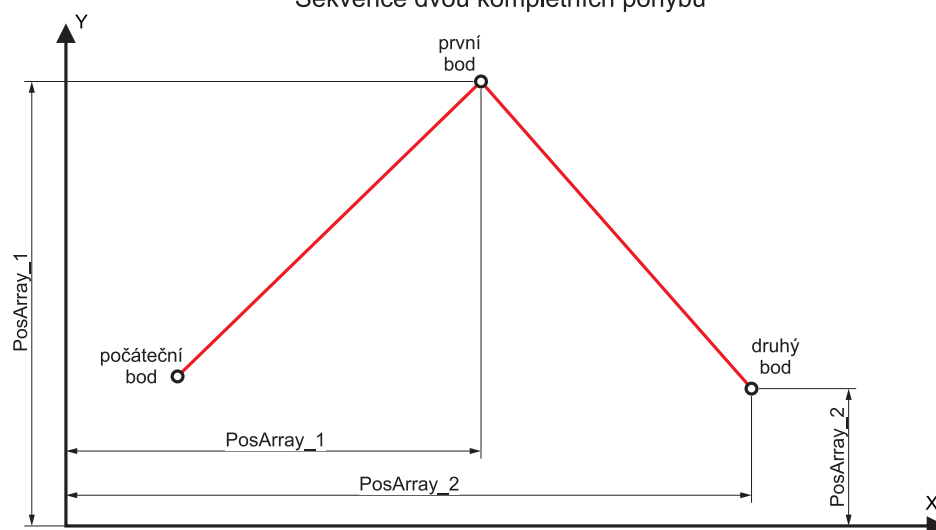
yAxesGroup	Odkaz na skupinu os	Reference
Done	Příznak dokončení algoritmu	Bool

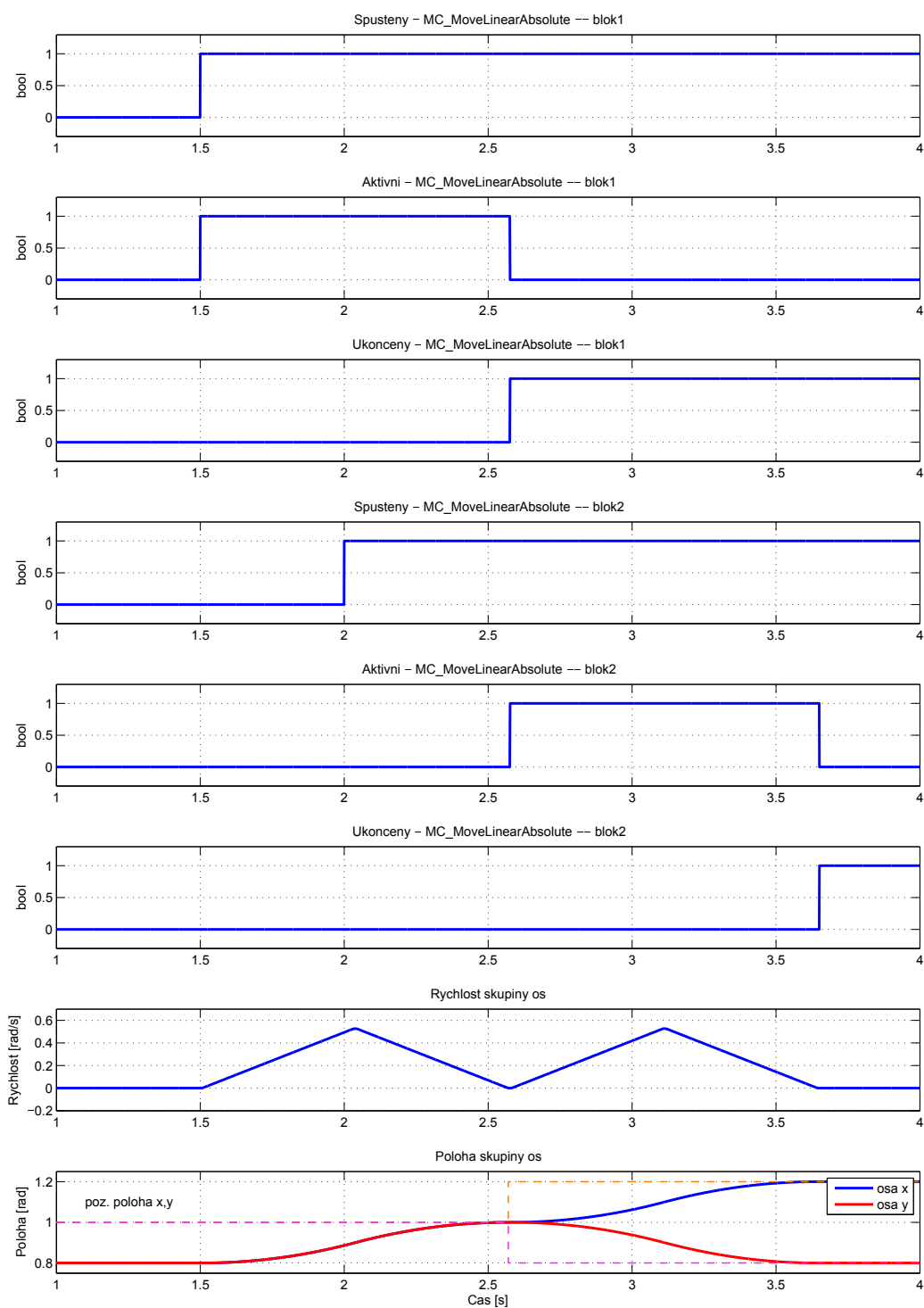
CommandAborted	Příznak přerušení funkce bloku	Bool
Busy	Příznak, že algoritmus ještě neskončil	Bool
Active	Příznak, že blok řídí osu	Bool
Error	Příznak chyby	Bool
ErrorID	Výsledek poslední operace	Error
i obecná chyba systému REXYGEN		

Příklad



Sekvence dvou kompletních pohybů

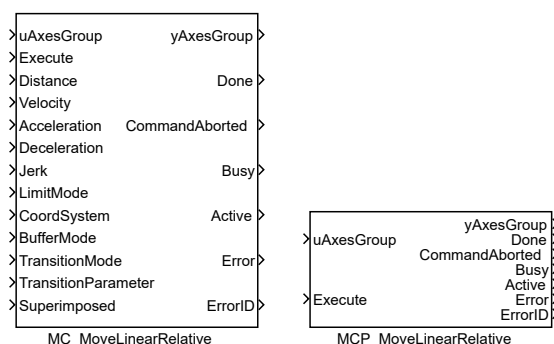




MC_MoveLinearRelative, MCP_MoveLinearRelative – Pohyb do pozice po přímkách (relativní souřadnice)

Symboly bloků

Licence: [COORDINATED MOTION](#)



Popis funkce

Bloky MC_MoveLinearRelative a MCP_MoveLinearRelative mají naprosto shodnou funkci, jediným rozdílem je, že MCP_ varianta bloku má méně vstupů a potřebné konstanty se zadávají jako parametry bloku.

Blok **MC_MoveLinearAbsolute** slouží pro přesun koncového efektoru po přímce na zadanou pozici. Pozice se zadává relativně od aktuální polohy v souřadném systému zvoleném vstupem **CoordSystem**. Parametry **Velocity**, **Acceleration**, **Deceleration** a **Jerk** určují rychlost, zrychlení, zpomalení a změnu zrychlení ve směru pohybu (tj. tečně k trajektorii). Pro určení těchto parametrů se vychází při použití souřadného systému MCS nebo PCS jen z polohových souřadnic. Další souřadnice (úhel natočení) se již generují proporcionálně, tak aby byl pohyb v těchto souřadnicích lineární a skončil ve stejném okamžiku jako polohové souřadnice. Pokud se poloha nemění (dochází tedy jen k otočení koncového efektoru), počítá se rychlost/zrychlení ze všech souřadnic, ale číslo má pak jiný fyzikální význam. Cílovou polohu určuje vektorový parametr **Distance**. Tento parametr musí mít tolik prvků, kolik předpokládá kinematická transformace (viz [MC_SetKinTransform_Lin](#)). V opačném případě je signalizována chyba a pohyb se neprovede. Pohyb je spuštěn náběžnou hranou na vstupu **Execute**.

Poznámka: Podle typu předchozího pohybu a v některých případech i podle parametrů (zejména v případě příliš krátké trajektorie) nemusí být implementované nebo realizovatelné všechny varianty **TransitionMode**.

Vstupy

uAxesGroup Odkaz na skupinu os

Reference

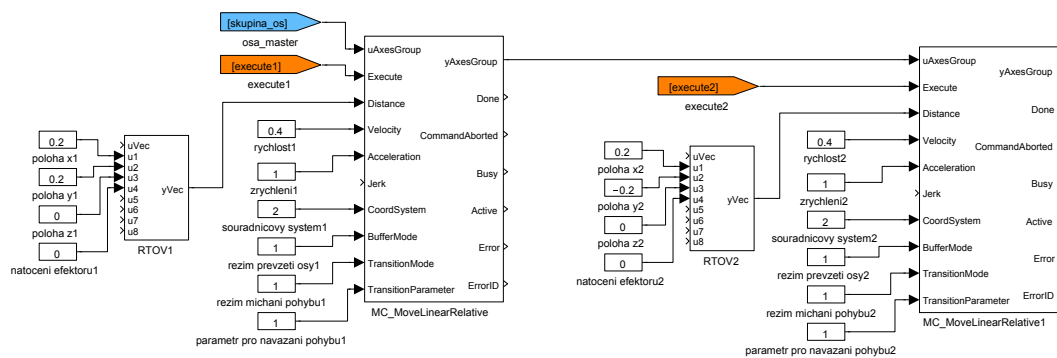
Execute	Náběžná hrana aktivuje blok	Bool
Distance	Pole souřadnic (relativních pozic a orientací)	Reference
Velocity	Maximální povolená rychlost [unit/s]	Double (F64)
Acceleration	Maximální povolené zrychlení [unit/s ²]	Double (F64)
Jerk	Maximální povolená změna zrychlení [unit/s ³]	Double (F64)
LimitMode	Volba jednotek pro limity (Velocity, Acceleration, Jerk) 1 Relative [part of group limit] 2 Absolute [unit/s, unit/s ² , unit/s ³]	⊙1 Long (I32)
CoordSystem	Volba souřadného systému 1 ACM 2 MCS 3 PCS	Long (I32)
BufferMode	Režim převzetí osy 1 Aborting (nový blok se spustí okamžitě) 2 Buffered (nový blok se spustí po dokončení předchozího) 3 Blending low (nový blok se spustí po dokončení předchozího, původní pohyb skončí s nižší rychlostí z obou bloků) 4 Blending high (nový blok se spustí po dokončení předchozího, původní pohyb skončí s vyšší rychlostí z obou bloků) 5 Blending previous (nový blok se spustí po dokončení předchozího, původní pohyb skončí se svojí koncovou rychlostí) 6 Blending next (nový blok se spustí po dokončení předchozího, původní pohyb skončí s počáteční rychlostí nového bloku)	Long (I32)
TransitionMode	Režim míchání pohybu 1 TMNone (xx) 2 TMstartvelocity (proložení s danou počáteční rychlostí) 3 TMConstantVelocity (proložení s danou konstantní rychlostí) 4 TMCornerDistance (xx) 5 TMMaxCornerDeviation (xx) 11 Smooth (nový blok se spustí po dokončení předchozího, původní pohyb skončí s počáteční rychlostí nového bloku)	Long (I32)
TransitionParameter	Parametr pro navázání pohybu (dle zvoleného režimu míchání)	Double (F64)
Superimposed	Příznak vykonání jako vedlejší (superimposed) pohyb	Bool

Výstupy

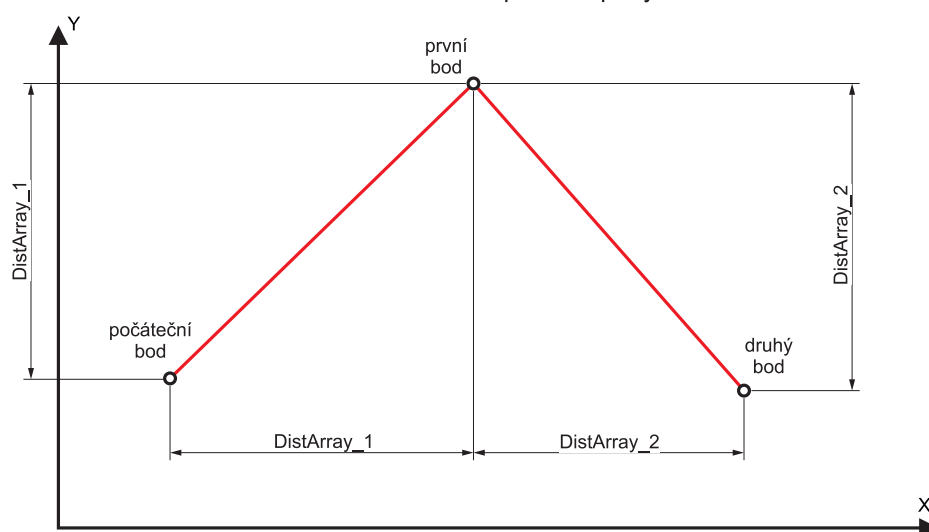
yAxesGroup	Odkaz na skupinu os	Reference
Done	Příznak dokončení algoritmu	Bool

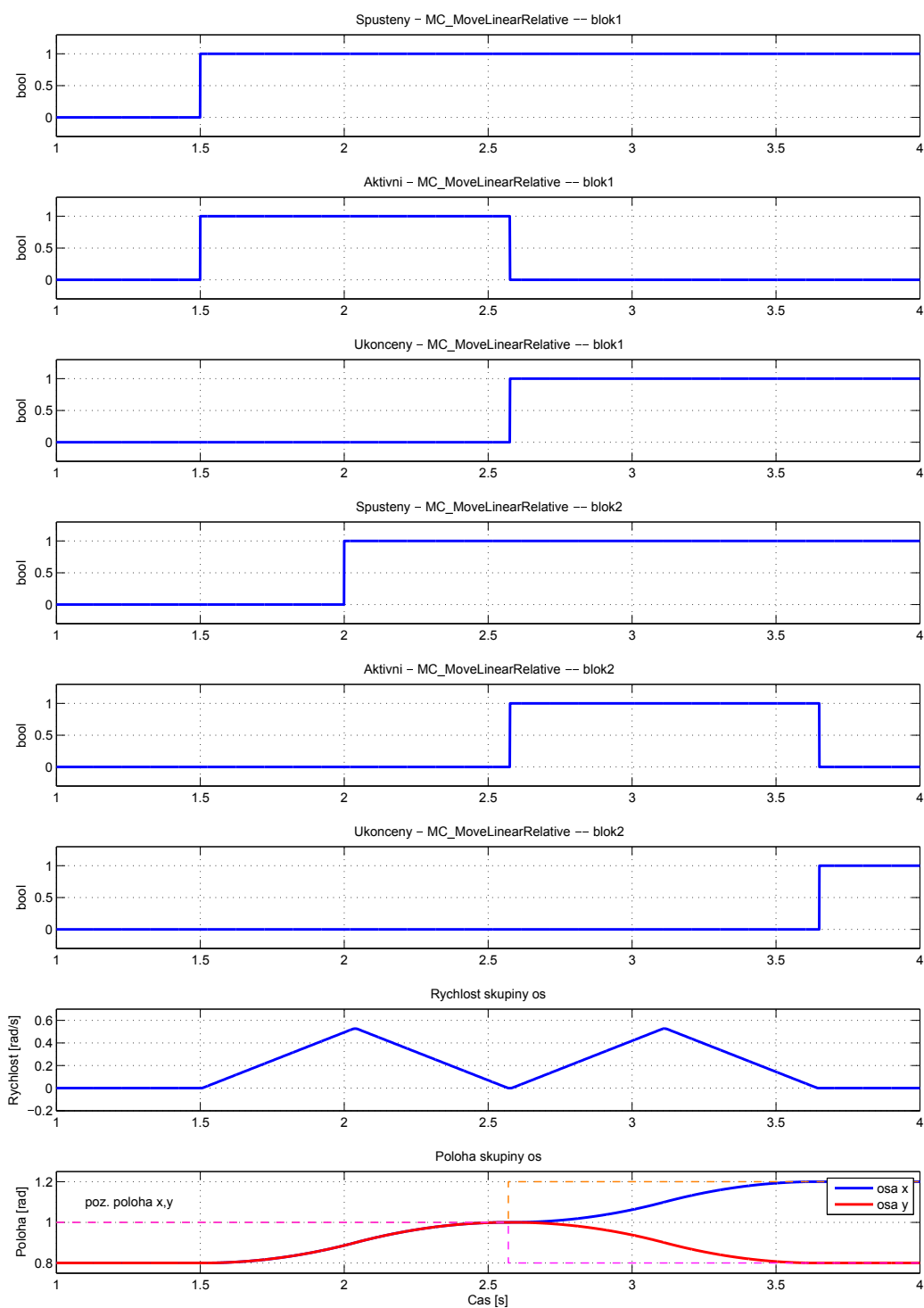
CommandAborted	Příznak přerušení funkce bloku	Bool
Busy	Příznak, že algoritmus ještě neskončil	Bool
Active	Příznak, že blok řídí osu	Bool
Error	Příznak chyby	Bool
ErrorID	Výsledek poslední operace	Error
i obecná chyba systému REXYGEN		

Příklad



Sekvence dvou kompletních pohybů

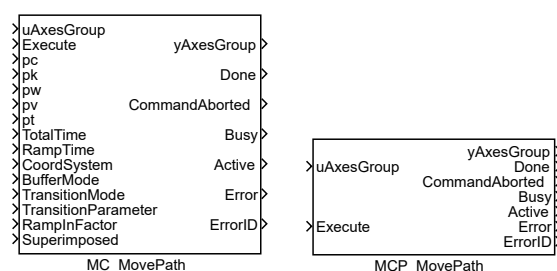




MC_MovePath, MCP_MovePath – Generování obecné trajektorie v prostoru

Symbyly bloků

Licence: **COORDINATED MOTION**



Popis funkce

Bloky MC_MovePath a MCP_MovePath mají naprosto shodnou funkci, jediným rozdílem je, že MCP_ varianta bloku má méně vstupů a potřebné konstanty se zadávají jako parametry bloku.

Blok MC_MovePath slouží pro pohyb stroje (skupiny os) po definované křivce. Křivka je zadána pomocí NURBS aproximace pátého řádu. Křivka je vždy absolutní, proto je před spuštěním tohoto bloku nutné nastavit stroj do počátku křivky. V opačném případě je hlášena chyba -707 (skok v poloze). Blok umožňuje zadat parametry NURBS aproximace i ručně, ale to je prakticky neproveditelné (snad jen jako import z jiného programu). K tomuto bloku existuje speciální editor trajektorie (spouští se tlačítkem **SpecialEdit** v dialogu pro zadávání parametrů bloku), kde se křivka zadává/nakreslí pomocí bodů v prostoru, kterými má procházet a NURBS parametry jsou pak napočítány automaticky. NURBS parametrizace definuje pouze tvar křivky (pokud jistý parametr probíhá od 0 do 1 pak výstupní hodnota/vektor probíhá požadovanou křivku). Pro pohyb stroje musíme ještě definovat rychlost, s jakou křivku prochází. Blok podporuje 2 způsoby. Buď zadáme celkovou dobu jízdy a popřípadě dobu zrychlování na začátku a na konci pomocí parametrů **TotalTime** a **RampTime** nebo vytvoříme rychlostní profil pomocí parametrů **pv** a **pt**. Tento profil se skládá z několika intervalů a v každém se aproximuje polynomem 5. stupně. Ruční vytváření profilu je prakticky neproveditelné a editor trajektorií to zatím nepodporuje. Nouzově je možné použít editor bloku MC_PositionProfile (kde mají data stejný formát), přičemž poloha musí začínat v 0 a končit v 1. Ve většině případů však vystačíme s první metodou, protože editor trajektorií generuje NURBS tak, aby pro konstantní rychlost změny vstupního parametru byla přibližně konstantní rychlost pohybu po křivce. Teorie NURBS řeší aproximaci funkce jedné proměnné. Protože poloha je vektor, potřebujeme několik funkcí jedné proměnné (nezávisle proměnná je pro

všechny tyto funkce parametr, který probíhá interval od 0 do 1). Teoreticky máme tedy pro každou souřadnici jednu NURBS funkci, nicméně použitá implementace má některé parametry společné pro všechny souřadnice (například řád, uzlové body). Ačkoliv v bloku lze použít souřadný systém ACS, grafický editor trajektorie předpokládá pravoúhlý souřadný systém. Taktéž jsou formálně dovoleny všechny buffered a transition režimy, ale vzhledem k povaze bloku je vhodné používat jen buffered nebo blending next (ostatní většinou vedou na chybu -707 - skok v poloze). Tento blok nemá vstup **PathData**, který vyžaduje specifikace **PLCopen**. V systému **REXYGEN** má tento blok všechna potřebná data (nebo odkazy na ně) uložena v parametrech. Z tohoto důvodu se také nepoužívá blok **MC_PathSelect**.

Vstupy

uAxesGroup	Odkaz na skupinu os	Reference
Execute	Náběžná hrana aktivuje blok	Bool
TotalTime	Čas celého pohybu	Double (F64)
RampTime	Čas [s] zrychlování/zpomalování	Double (F64)
CoordSystem	Volba souřadného systému	Long (I32)
	1 ACM	
	2 MCS	
	3 PCS	
BufferMode	Režim převzetí osy	Long (I32)
	1 Aborting (nový blok se spustí okamžitě)	
	2 Buffered (nový blok se spustí po dokončení předchozího)	
	3 Blending low (nový blok se spustí po dokončení předchozího, původní pohyb skončí s nižší rychlostí z obou bloků)	
	4 Blending high (nový blok se spustí po dokončení předchozího, původní pohyb skončí s vyšší rychlostí z obou bloků)	
	5 Blending previous (nový blok se spustí po dokončení předchozího, původní pohyb skončí se svojí koncovou rychlostí)	
	6 Blending next (nový blok se spustí po dokončení předchozího, původní pohyb skončí s počáteční rychlostí nového bloku)	
TransitionMode	Režim míchání pohybu	Long (I32)
	1 TMNone (xx)	
	2 TMstartvelocity (proložení s danou počáteční rychlostí)	
	3 TMConstantVelocity (proložení s danou konstantní rychlostí)	
	4 TMCornerDistance (xx)	
	5 TMMaxCornerDeviation (xx)	
	11 Smooth (nový blok se spustí po dokončení předchozího, původní pohyb skončí s počáteční rychlostí nového bloku)	

TransitionParameter	Parametr pro navázání pohybu (dle zvoleného režimu míchání)	Double (F64)
RampInFactor	faktor pro najíždění na počáteční polohu (0 = RampIn režim se nepoužívá), odpovídá přibližně relativní rychlosti a zrychlení	Double (F64)
Superimposed	Příznak vykonání jako vedlejší (superimposed) pohyb	Bool

Parametry

pc	Matice řídicích bodů $\odot[0.0 \ 1.0 \ 2.0; \ 0.0 \ 1.0 \ 1.0; \ 0.0 \ 1.0 \ 0.0]$	Double (F64)
pk	Uzlový vektor $\odot[0.0 \ 0.0 \ 0.0 \ 0.0 \ 0.5 \ 1.0 \ 1.0]$	Double (F64)
pw	Váhový vektor $\odot[1.0 \ 1.0 \ 1.0]$	Double (F64)
pv	Polynom pro korekci rychlosti posuvu $\odot[0.0 \ 0.05 \ 0.95; \ 0.0 \ 0.1 \ 0.1; \ 0.0 \ 0.0 \ 0.0; \ 0.1 \ 0.0 \ -0.1; \ -0.05 \ 0.0 \ 0.05; \ 0.0 \ 0.0 \ 0.0]$	Double (F64)
pt	Uzlové body pro korekci posuvu $\odot[0.0 \ 1.0 \ 10.0 \ 11.0]$	Double (F64)
user	Pouze pro special edit $\odot[0.0 \ 1.0 \ 2.0 \ 3.0]$	Double (F64)

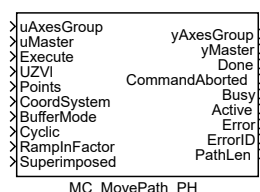
Výstupy

yAxesGroup	Odkaz na skupinu os	Reference
Done	Příznak dokončení algoritmu	Bool
CommandAborted	Příznak přerušení funkce bloku	Bool
Busy	Příznak, že algoritmus ještě neskončil	Bool
Active	Příznak, že blok řídí osu	Bool
Error	Příznak chyby	Bool
ErrorID	Výsledek poslední operace i obecná chyba systému REXYGEN	Error

MC_MovePath_PH – * Generování obecné trajektorie v prostoru PH

Symbol bloku

Licence: [COORDINATED MOTION](#)



Popis funkce

Popis tohoto bloku ještě není k dispozici. Níže naleznete částečný popis vstupů, výstupů a parametrů bloku. Kompletní popis bloku bude k dispozici v dalších revizích dokumentace.

Tento blok nepropaguje kvalitu signálu. Více informací je uvedeno v sekci [1.4](#).

Vstup

uAxesGroup	Odkaz na skupinu os	Reference
uMaster	Odkaz na osu (přípustné je jen spojení RM_Axis.axisRef-uAxis nebo yAxis-uAxis)	Reference
Execute	Náběžná hrana aktivuje blok	Bool
CoordSystem	Volba souřadného systému	Long (I32)
	1 ACS	
	2 MCS	
	3 PCS	
	4 TCS	
BufferMode	Režim převzetí osy	Long (I32)
	1 aborting	
	2 buffered	
	3 blending low	
	4 blending high	
	5 blending previous	
	6 blending next	
Cyclic	Příznak cyklicky opakovaného profilu	Bool
RampIn	RampIn faktor (0 = RampIn režim se nepoužívá)	Double (F64)
UZVI		Reference
Points		Reference

Parametr

Superimposed	příznak spuštění jako aditivní pohyb k existujícímu	Bool

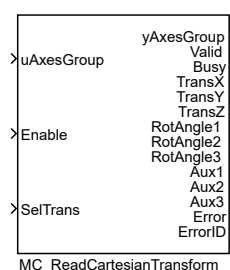
Výstup

<code>yAxesGroup</code>	Odkaz na skupinu os	Reference
<code>yMaster</code>	Odkaz na osu (přípustné je jen spojení <code>RM_Axis.axisRef-uAxis</code> nebo <code>yAxis-uAxis</code>)	Reference
<code>Done</code>	Příznak dokončení algoritmu	Bool
<code>CommandAborted</code>	Příznak přerušení funkce bloku	Bool
<code>Busy</code>	Příznak, že algoritmus ještě neskončil	Bool
<code>Active</code>	Příznak, že blok řídí osu	Bool
<code>Error</code>	Příznak chyby	Bool
<code>ErrorID</code>	Výsledek poslední operace	Error
<code>PathLen</code>		Double (F64)

MC_ReadCartesianTransform – Přechzení použité kartézské transformace

Symbol bloku

Licence: [COORDINATED MOTION](#)



Popis funkce

Blok `MC_ReadCartesianTransform` zpřístupňuje na výstup `TransX`, `TransY`, `TransZ`, `RotAngle1`, `RotAngle2`, `RotAngle3`, `Aux1`, `Aux2` a `Aux3` aktivní kartézskou transformaci mezi souřadnými systémy. V systému lze definovat několik kartézských transformací. Kterou chceme zobrazit se vybírá parametrem `SelTrans`. Význam hodnot je vysvětlen v popisu bloku `MC_SetCartesianTransform`. Hodnota je platná jen pokud je na výstupu `Valid` true, čehož se dosáhne nastavením vstupu `Enable` na hodnotu true. Pokud je aktivní více než jedna kartézská transformace, tak je na výstupu dána výsledná kartézská transformace.

Vstupy

<code>uAxesGroup</code>	Odkaz na skupinu os	Reference
<code>Enable</code>	Povolení funkce bloku (aktivace výstupů)	Bool
<code>SelTrans</code>	Výběr transformace pro zobrazení	⊙1 Long (I32)
	1 PCS offset	
	2 Tool offset	
	3 Machine Base offset	

Výstupy

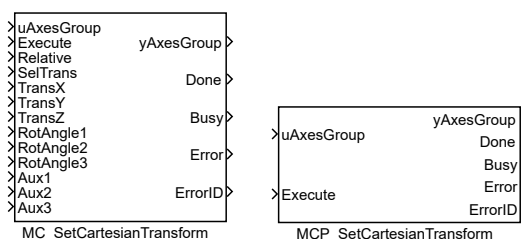
<code>yAxesGroup</code>	Odkaz na skupinu os	Reference
<code>Valid</code>	Příznak platnosti výstupní hodnoty	Bool
<code>Busy</code>	Příznak, že algoritmus ještě neskončil	Bool
<code>TransX</code>	Posun v ose X	Double (F64)
<code>TransY</code>	Posun v ose Y	Double (F64)
<code>TransZ</code>	Posun v ose Z	Double (F64)
<code>RotAngle1</code>	Úhel rotace podél osy X	Double (F64)

RotAngle2	Úhel rotace podél osy Y	Double (F64)
RotAngle3	Úhel rotace podél osy Z	Double (F64)
Aux1	posun v 1. doplňkové ose	Double (F64)
Aux2	posun v 2. doplňkové ose	Double (F64)
Aux3	posun v 3. doplňkové ose	Double (F64)
Error	Příznak chyby	Bool
ErrorID	Výsledek poslední operace i obecná chyba systému REXYGEN	Error

MC_SetCartesianTransform, MCP_SetCartesianTransform – Kartézská transformace

Symbole bloků

Licence: [COORDINATED MOTION](#)



Popis funkce

Bloky MC_SetCartesianTransforms a MCP_SetCartesianTransforms mají naprosto shodnou funkci, jediným rozdílem je, že MCP_ varianta bloku má méně vstupů a potřebné konstanty se zadávají jako parametry bloku.

Blok **MC_SetCartesianTransform** nastavuje kartézskou transformaci mezi souřadnými systémy. V systému lze definovat několik kartézských transformací. Kterou chceme nastavit se vybírá parametrem **SelTrans**. Pomocí vstupů **Trans** se nastavuje posunutí v jednotlivých osách. Pomocí vstupů **RotAngle** se pak nastavuje rotace kolem daných os (pořadě osa Z, osa Y, osa X). Pomocí vstupů **Aux** se nastavuje transformace dodatečných os, pokud je systém/kinematická transformace používá (transformace je přičtení v jednotlivých aux souřadnicích).

Vstupy

		Reference
uAxesGroup	Odkaz na skupinu os	
Execute	Náběžná hrana aktivuje blok	Bool
Relative	Výběr absolutní (=false) nebo relativní (=true) transformace; absolutní transformace nastaví transformační matici podle nových hodnot, relativní transformace přidá nové hodnoty již nastavené transformaci (tj. vynásobí původní a novou transformací matici)	Bool
SelTrans	Výběr transformace pro nastavení	⊙1 Long (I32)
	1 PCS offset; transformace mezi základnou (WCS) a výrobkem (PCS)	
	2 Tool offset; transformace mezi koncem robota a středem nástroje	
	3 Machine Base offset; transformace mezi základnou (WCS) a patou robota/stroje (MCS)	

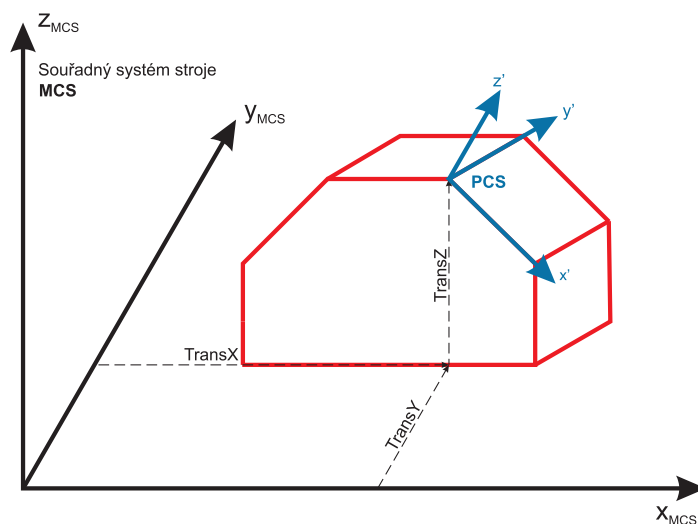
TransX	Posun v ose X	Double (F64)
TransY	Posun v ose Y	Double (F64)
TransZ	Posun v ose Z	Double (F64)
RotAngle1	Úhel rotace podél osy Z	Double (F64)
RotAngle2	Úhel rotace podél osy Y	Double (F64)
RotAngle3	Úhel rotace podél osy X	Double (F64)
Aux1	posun v 1. doplňkové ose	Double (F64)
Aux2	posun v 2. doplňkové ose	Double (F64)
Aux3	posun v 3. doplňkové ose	Double (F64)

Výstupy

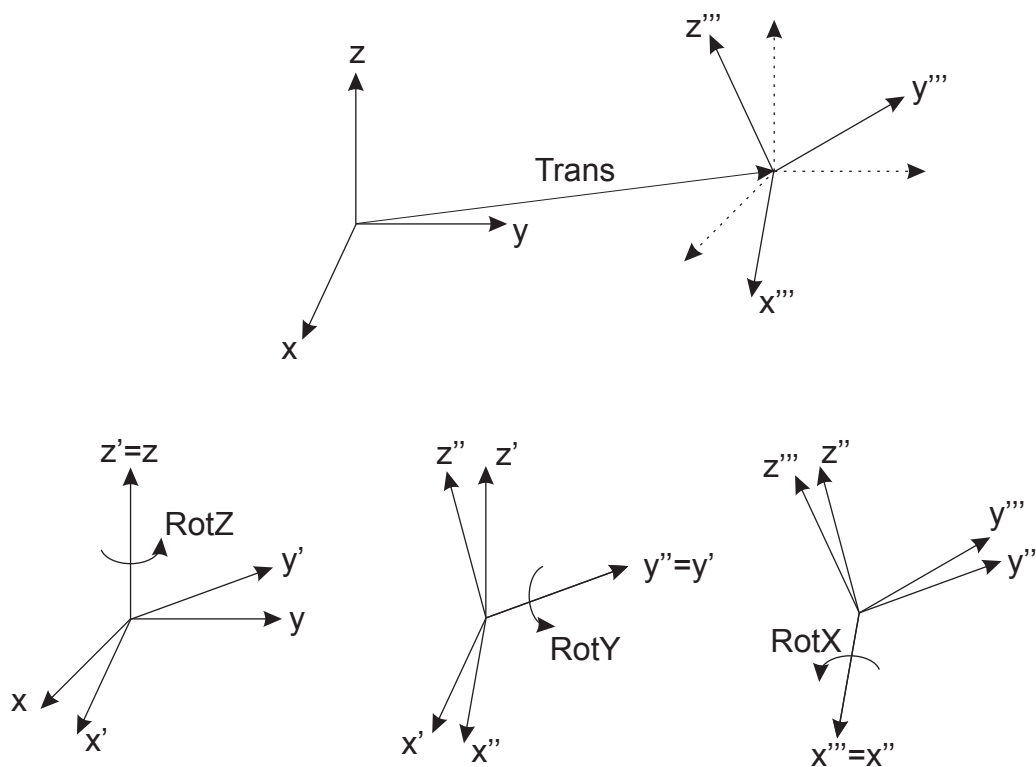
yAxesGroup	Odkaz na skupinu os	Reference
Done	Příznak dokončení algoritmu	Bool
Busy	Příznak, že algoritmus ještě neskončil	Bool
Error	Příznak chyby	Bool
ErrorID	Výsledek poslední operace i obecná chyba systému REXYGEN	Error

Definice translace a rotace

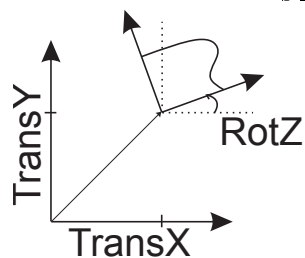
Na prvním obrázku je uveden příklad translace ze souřadného systému MCS do souřadného systému PCS.



Na druhém obrázku je pak uveden příklad rotace. Kdy celková rotace je docílena postupným provedením dílčích rotací kolem jednotlivých os.



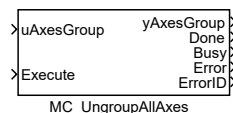
Ukázka zadání kartézské transformace (v rovině). Pro dosažení transformace na obrázku níže se spustí blok `MC_SetCartesianTransform` s nastavenou transformací $\{50,50,0,30,0,0\}$



MC_UngroupAllAxes – Odebrání všech os ze skupiny

Symbol bloku

Licence: [COORDINATED MOTION](#)



Popis funkce

Blok **MC_UngroupAllAxes** odebere všechny osy ze dané skupiny os. Po dokončení je změněn stav skupiny na „GroupDisabled“. Blok kromě odebrání os odebere i kinematickou transformaci, kartézské transformace a všechna další nastavení, tj. převede stav bloku [RM_AxisGroup](#) do počátečního stavu.

Poznámka 1: Pokud je blok spuštěn a skupina os není ve stavu „GroupDisabled“, „GroupStandBy“ nebo „GroupErrorStop“, tak je vyvolána chyba a blok není spuštěn.

Vstupy

uAxesGroup	Odkaz na skupinu os	Reference
Execute	Náběžná hrana aktivuje blok	Bool

Výstupy

yAxesGroup	Odkaz na skupinu os	Reference
Done	Příznak dokončení algoritmu	Bool
Busy	Příznak, že algoritmus ještě neskončil	Bool
Error	Příznak chyby	Bool
ErrorID	Výsledek poslední operace	Error
	i obecná chyba systému REXYGEN	

RM_AxesGroup – Skupina os pro koordinované řízení pohybu

Symbol bloku

Licence: [COORDINATED MOTION](#)



Popis funkce

Blok **RM_AxesGroup** je základní blok skupiny os pro koordinovaný pohyb. Představuje sdílenou strukturu, kde jsou uloženy všechny stavy a parametry skupiny. Algoritmus tohoto bloku představuje kontrolu nastavených mezí, havarijní zastavení v případě potřeby, předávání dat do a z bloků **RM_Axis** podřízených os, přepočítání všech stavů a výstupů pro případ, že žádný blok není aktivní, ale je potřeba generovat pohyb (např. zastavení v PCS a částečně i algoritmus pro generování požadované trajektorie). Výstupem tohoto bloku jsou pouze stavové a pomocné signály (viz dále).

Před spuštěním koordinovaného pohybu je nutné ke skupině (bloku **RM_AxesGroup**) připojit jednotlivé osy (spuštěním bloku **MC_AddAxisToGroup** pro každou osu), přiřadit kinematickou transformaci (spuštěním bloku jednoho z bloků **MC_SetKinTransform_Xxx**) a skupinu povolit (spuštěním bloku **MC_GroupEnable**).

Další důležité administrativní bloky pro řízení stavu skupiny os jsou **MC_GroupReset** (smaže chybu, tj. přepíná skupinu ze stavu **ErrorStop** do stavu **Standstill**), **MC_GroupDisable** (přepíná skupinu os do stavu **PowerOff**) a **MC_UngroupAllAxis** (ruší všechna nastavení, tj. odebere osy, odebere kinematickou transformaci, přepne do stavu **PowerOff**).

Implicitní hodnoty parametrů (zejména limity na rychlost a zrychlení) jsou záměrně nastaveny na 0, což je nedovolená hodnota. Všechny parametry tak musí nastavit uživatel podle skutečných možností připojeného motoru a stroje.

Víme, že blok **RM_AxisSpline** může být v rychlejší úloze než blok **RM_Axis** a pak se musí provádět interpolace polohy a rychlosti. Podobně bloky **RM_Axis** mohou být v rychlejší úloze než **RM_AxesGroup** a pak je potřeba provádět interpolaci. Interpolaci je potřeba provádět v rychlejší úloze (tj. v blocích **RM_Axis**), ale režim interpolace se zadává v bloku **RM_AxesGroup** parametrem **InterpolationMode**.

Vektortové parametry **min** a **max** jsou nepovinné. Pokud jsou zadány, zadávají se ve stejném formátu jako poloha skupiny (tj. např. parametr **Position** bloku **MC_MoveLinearAbsolute**). Tento formát je definován kinematickou transformací, tj. blokem **MC_SetKinTransform_Lin** nebo podobným (tj. **MC_SetKinTransform_Xxx**). Pohyb je zastaven (a blok **RM_AxesGroup** přejde do stavu **ErrorStop**, pokud hrozí překročení hodnoty v kterékoliv souřadnici. Rotační souřadnice se ale nekontrolují.

Poznámka: U omezení na rychlost otáčení (parametr **VelRot**) je základní jednotka radián za sekundu. Připojená kinematická transformace však může změnit jednotku pro

zadávaní úhlu (např. na úhlové stupně nebo otáčky) a pak se rychlost zadává v těchto jednotkách za sekundu. Stejně pravidlo platí pro zrychlení a jerk, tj. parametry **AccRot** a **JerkRot**.

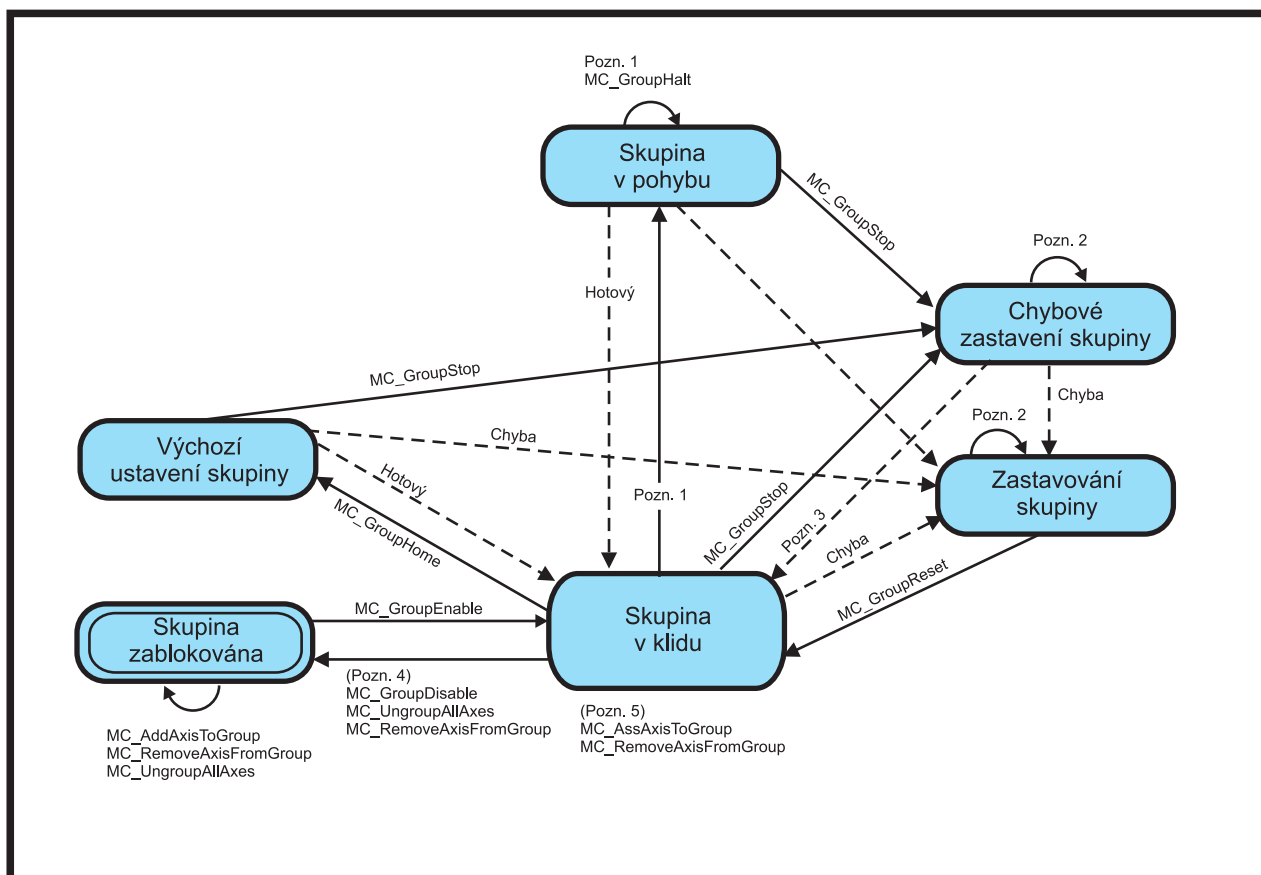
Parametry

Velocity	Maximální povolená rychlost [unit/s]	Double (F64)
Acceleration	Maximální povolené zrychlení [unit/s ²]	Double (F64)
Jerk	Maximální povolená změna zrychlení [unit/s ³]	Double (F64)
VelRot	Maximální povolená rychlost otáčení [rotunit/s]	Double (F64)
AccRot	Maximální povolené zrychlení v otáčení [rotunit/s ²]	Double (F64)
JerkRot	Maximální povolená změna zrychlení v otáčení [rotunit/s ³]	Double (F64)
InterpolationMode	Algoritmus pro interpolaci	Long (I32)
	1 linear (poloha interpolována lineárně, rychlost jako derivace polohy, zrychlení 0, tj. rychlost je po částech konstantní funkce se skoky)	
	2 cubic spline (poloha je polynom 3. řádu vypočtený na základě polohy a rychlosti na začátku a konci intervalu, rychlost je derivace polohy, zrychlení derivace rychlosti)	
	3 quintic spline (poloha je polynom 5. řádu vypočtený na základě polohy, rychlosti a zrychlení na začátku a konci intervalu, rychlost je derivace polohy, zrychlení derivace rychlosti)	
	4 cubic Bspline approximation (poloha je polynom 3. řádu vypočtený na základě dvou poloh před a dvou poloh za aktuálním intervalem, proložená funkce nemusí přesně procházet zadanými body, rychlost je derivace polohy, zrychlení derivace rychlosti)	
	5 quintic Bspline approximation (poloha je polynom 5. řádu vypočtený na základě tří poloh před a tří poloh za aktuálním intervalem, proložená funkce nemusí přesně procházet zadanými body, rychlost je derivace polohy, zrychlení derivace rychlosti)	
	6 all linear (poloha, rychlost i zrychlení jsou interpolovány lineárně navzájem nezávisle, tj. rychlost neodpovídá přesně derivaci polohy a zrychlení neodpovídá přesně derivaci rychlosti)	
	7 all cubic (poloha i rychlost jsou interpolovány polynomem 3. řádu navzájem nezávisle, tj. rychlost neodpovídá přesně derivaci polohy)	
	8 rezervováno pro pozdější použití	
	9 rezervováno pro pozdější použití	
min	Minimální poloha v MCS (vektor). Při překročení kterékoliv souřadnice je pohyb zastaven a blok přejde do stavu ErrorStop	Double (F64)
		⊙[]
max	Maximální poloha v MCS (vektor). Při překročení kterékoliv souřadnice je pohyb zastaven a blok přejde do stavu ErrorStop	Double (F64)
		⊙[]

Výstupy

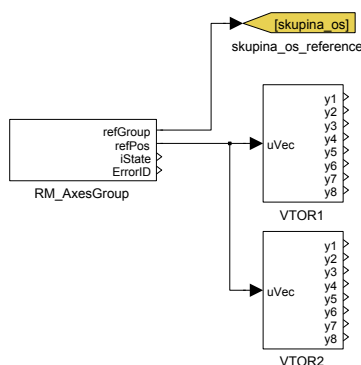
refGroup	Odkaz na skupinu os	Reference
refPos	Vektor generovaných poloh a rychlostí v MCS	Reference
iState	Stav skupiny	Long (I32)
	0 Disabled (skupina blokována)	
	1 Standby (skupina připravena)	
	2 Homing (hledání výchozí polohy)	
	6 Moving (koordinovaný pohyb)	
	7 Stopping (zastavování nebo dočasné blokování)	
	8 Error stop (zastavování nebo blokování skupiny po chybě)	
ErrorID	Výsledek poslední operace	Error
	i obecná chyba systému REXYGEN	

Stavový diagram skupiny os



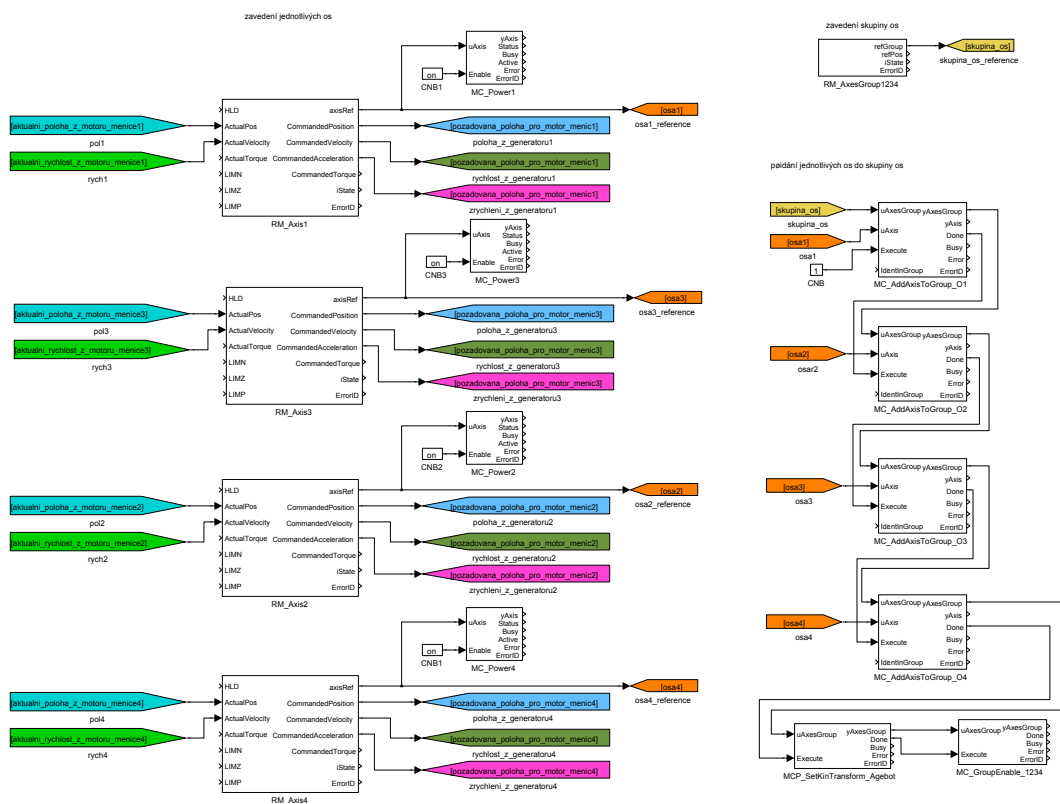
Čtení z výstupu refPos

Z výstupu **refPos** bloku **RM_AxesGroup** lze číst aktuální polohu a rychlost (stejně jako pomocí bloků **MC_GroupReadActualPosition**, **MC_GroupReadActualVelocity**; zrychlení zde není vyčíslováno a musí se použít **MC_GroupReadActualAcceleration**). Výstup **refPos** je vektorový, pro získání konkrétních hodnot je nutné použít blok **VTOR** z knihovny **MA-TRIX**. Tento blok má 8 výstupů. Pokud je souřadnic více než 8, vyřeší se to paralelním připojením dalšího bloku **VTOR** (viz. Obr. níže) do kterého je nutné zadat jako parametr správný offset (od jakého indexu se mají dát data na výstup). Např. robot AGEBOT (jeho sestavení jako skupiny os viz. níže) má čtyři osy. Bude mít tedy čtyři souřadnice pro polohu a rychlost. Jedním blokem **VTOR** získáme informaci o poloze a rychlosti (výstupy $y_1 - y_4$ a $y_5 - y_8$ bloku **VTOR1**).



Sestavení skupiny os

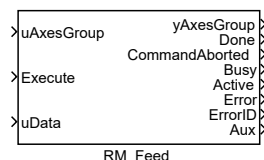
Pro správné spuštění bloku generujícího víceosý pohyb je nutné definovat jednotlivé osy **RM_Axis** a postupně je všechny přidat do skupiny os pomocí bloku **MC_AddAxisToGroup**. Dále je třeba nastavit kinematickou transformaci (dle typu stroje jeden z bloků **MC_SetKinTransform_XXX**). Nakonec je třeba aktivovat skupinu os blokem **MC_GroupEnable**. Pro pohyby v PCS je nutné ještě přidat kartézskou transformaci (blok **MC_SetCartesianTransforms**). Zpět do výchozího stavu (tj. před přiřazení os do skupiny) se skupina dostane spuštěním bloku **MC_UngroupAllAxis**. Příklad sestavení osy je uveden na obrázku níže.



RM_Feed – MC „krmič“

Symbol bloku

Licence: **COORDINATED MOTION**



Popis funkce

Tento blok slouží pro předání předem vygenerované trajektorie do infrastruktury pro koordinované řízení pohybu. Data se předávají pomocí matice přivedené na vstup **uData**. V základním režimu, kdy **VelFactor=1** je v každé periodě úlohy nastavena hodnota z dalšího sloupce matice. Hodnoty musí být ve sloupci v pořadí: číslo předávané na výstup **aux**, vektor polohy, dále nepovinně vektor rychlosti, nepovinně vektor zrychlení, nepovinně vektor sil/momentů. Pokud vektor rychlosti a/nebo zrychlení není zadán, dopočítává se diferencí. Momenty v celém systému řízení pohybu slouží jen jako dopředná vazba pro zpětnovazební regulátor servoměniče, takže pokud není znám generuje se 0 (což ostatně dělá i většina ostatních pohybových bloků). Vektor polohy/rychlosti/zrychlení se zadává ve stejném tvaru jako pro ostatní pohybové bloky, např. **MC_MoveLinearAbsolute**. Pokud **VelFactor** není 1, hodnoty polohy/rzchlosti/zrychlení jsou interpolovány lineárně, přičemž rychlost a zrychlení musí být násobeno příslušným faktorem (aby rychlost stále odpovídala derivaci polohy a zrychlení derivaci rychlosti).

Význam ostatních parametrů je stejný jako pro jiné pohybové bloky. Z principu funkce však musí být odstartován ze správné polohy, jinak dojde k chybě -707 (skok v poloze).

Vstupy

uAxesGroup	Odkaz na skupinu os	Reference
Execute	Náběžná hrana aktivuje blok	Bool
uData	Odkaz na matici s daty	Reference

Parametry

VelFactor	Faktor rychlosti. Pro nominální rychlost (tak jak byla data vygenerována) volíme 1, pro zpomalení pohybu čísla menší než 1 a pro zrychlení čísla větší než 1	Double (F64) ↓0.01 ↑100.0 ⊙1.0
Relative	Pokud je zatrženo, poloha v datech se přičte k poloze na startu (tj. první sloupec v matici by měl být 0)	Bool

CoordSystem	Volba souřadného systému	↓1 ↑3 ⊙2	Long (I32)
	1 ACM		
	2 MCS		
	3 PCS		
BufferMode	Režim převzetí osy	↓1 ↑6 ⊙1	Long (I32)
	1 Aborting (nový blok se spustí okamžitě)		
	2 Buffered (nový blok se spustí po dokončení předchozího)		
	3 Blending low (nový blok se spustí po dokončení předchozího, původní pohyb skončí s nižší rychlostí z obou bloků)		
	4 Blending high (nový blok se spustí po dokončení předchozího, původní pohyb skončí s vyšší rychlostí z obou bloků)		
	5 Blending previous (nový blok se spustí po dokončení předchozího, původní pohyb skončí se svojí koncovou rychlostí)		
	6 Blending next (nový blok se spustí po dokončení předchozího, původní pohyb skončí s počáteční rychlostí nového bloku)		
TransitionMode	Režim míchání pohybu	↓0 ↑15 ⊙1	Long (I32)
	1 TMNone (xx)		
	2 TMstartvelocity (proložení s danou počáteční rychlostí)		
	3 TMConstantVelocity (proložení s danou konstantní rychlostí)		
	4 TMCornerDistance (xx)		
	5 TMMaxCornerDeviation (xx)		
	11 Smooth (nový blok se spustí po dokončení předchozího, původní pohyb skončí s počáteční rychlostí nového bloku)		
TransitionParameter	Parametr pro navázání pohybu (dle zvoleného režimu míchání)		Double (F64)

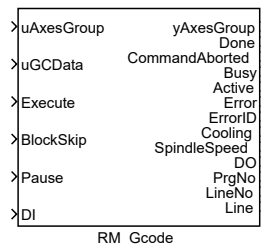
Výstupy

yAxesGroup	Odkaz na skupinu os	Reference
Done	Příznak dokončení algoritmu	Bool
CommandAborted	Příznak přerušení funkce bloku	Bool
Busy	Příznak, že algoritmus ještě neskončil	Bool
Active	Příznak, že blok řídí osu	Bool
Error	Příznak chyby	Bool
ErrorID	Výsledek poslední operace	Error
Aux	První hodnota v datech pro pomocné účely	Double (F64)

RM_Gcode – CNC řízení pohybu

Symbol bloku

Licence: **COORDINATED MOTION**



Popis funkce

G-code původně vznikl pro řízení obráběcích strojů pomocí počítače kolem roku 1950 (tzv. technologie NC a CNC). Tehdy se zadával pomocí děrné pásky nebo děrného štítku, proto je poměrně úsporný. V současnosti je to stále dominantní způsob zadávání sekvence pohybu (tj. programu) pro obráběcí stroje, ale i některé další typy strojů, např. řezacích plotrů, 3D tiskáren. Ačkoliv byl **G-code** standardizován již v 60. letech s drobnými úpravami v 80. letech, výrobci strojů používají různá rozšíření, aby bylo možné využít schopnosti jejich stroje. Dále jsou drobné rozdíly podle typu stroje (frézy mají některé příkazy odlišné od soustruhů nebo 3D tiskáren). Základní struktura příkazu/parametru je sekvence písmeno následovaná číslem. Příkazy na jedné řádce se považují za zadané současně, tj. nezávisí na pořadí. Příkazy/parametry, které na řádce nejsou se považují za nezměněné od minulého výskytu. Další informace k **G-code** např. <https://en.wikipedia.org/wiki/G-code>. V dnešní době se **G-code** prakticky vždy generuje z CAD programu.

V základní podobě se soubory s **G-code** umísťují do adresáře určeného parametrem **BaseDir** na cílové platformě. Soubory musí mít název <čtyřciferné číslo>.nc. Číslo programu, který se má spustit se pak zadává v parametru **MainFile**, přičemž program může volat další podprogramy (v **G-code** se identifikují číslem, tj. soubor ve stejném formátu i ve stejném adresáři jako hlavní program). Např. pokud **MainFile** = 1, otevírá se soubor 0001.nc.

Druhá možnost je na vstup **uGCData** připojit speciální blok, který generuje řádky **G-code** z jiných dat.

Vstupy

uAxesGroup	Odkaz na skupinu os	Reference
uGCData	Odkaz na speciální blok generující G-code	Reference
Execute	Náběžná hrana aktivuje blok	Bool
BlockSkip	Přeskočení bloku v G-code (pokud program přijde na řádku, kde je znak lomítka a je BlockSkip =true, tak se řádka přeskočí)	Bool

<code>workOffsets</code>	Sady počátečních souřadnic	$\odot$ [0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]	Double (F64)
<code>toolOffsets</code>	Sady korekcí délky nástroje	$\odot$ [0 0 0]	Double (F64)
<code>cutterOffsets</code>	Poloměry nástroje	$\odot$ [0 0 0]	Double (F64)

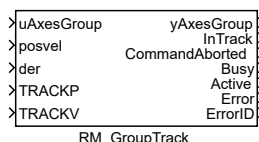
Výstupy

<code>yAxesGroup</code>	Odkaz na skupinu os	Reference
<code>Done</code>	Příznak dokončení algoritmu	Bool
<code>CommandAborted</code>	Příznak přerušení funkce bloku	Bool
<code>Busy</code>	Příznak, že algoritmus ještě neskončil	Bool
<code>Active</code>	Příznak, že blok řídí osu	Bool
<code>Error</code>	Příznak chyby	Bool
<code>ErrorID</code>	Výsledek poslední operace i obecná chyba systému REXYGEN	Error
<code>Cooling</code>	Chlazení zapnuto	Bool
<code>SpindleSpeed</code>	Rychlost otáčení vřetene	Double (F64)
<code>D0</code>	Celé číslo, které slouží jako bitové pole výstupních logických signálů, které lze v G-code nastavovat	Long (I32)
<code>PrgNo</code>	Číslo právě vykonávaného programu	Long (I32)
<code>LineNo</code>	Číslo právě vykonávané řádky programu (parametr N v G-code)	Long (I32)
<code>Line</code>	Právě prováděná řádka G-kódu	String

RM_GroupTrack – * Sledování polohy/rychlosti

Symbol bloku

Licence: [COORDINATED MOTION](#)



Popis funkce

Blok **RM_GroupTrack** slouží pro sledování externího signálu. Je to obdoba bloku **RM_Track** pro koordinovaný pohyb. Blok se aktivuje nastavením vstupu **TRACKP** = 1 nebo **TRACKV** = 1. V prvním případě se blok snaží co nejrychleji (ovšem s respektováním maximální zadané rychlosti, zrychlení, popřípadě jerku nastavené parametry **Velocity**, **Acceleration**, **Jerk**) dosáhnout polohy přivedené na vstup **posvel**. Poloha se může v každém kroku algoritmu měnit a blok se stále přepočítává trajektorii a snaží se sledovat vstupní signál. Vstupní vektor má stejný formát jako poloha v bloku **MC_MoveLinearAbsolute** a podobných (tj. přesený formát definuje kinematická transformace). Ve druhém případě se na vstup **posvel** přivádí vektor rychlosti a blok se opět snaží řídit skupinu os, aby co nej přesněji odpovídala požadované hodnotě, ale byly respektovány zadaná omezení. Na vstup **der** se přivádí derivace signálu přivedeného na **posvel** (tj. v prvním režimu to odpovídá rychlosti, ve druhém zrychlení). Protože poloha/rychlost je vektorová veličina, je i její derivace vektor. Pokud hodnota derivace není známa, ponechává se vstup nepřipojen a derivace je odhadována z předchozího průběhu signálu (diferencí z několika posledních hodnot podle parametru **iLen**). Pokud je ale derivace známa, je vhodné ji připojit, protože to zlepšuje přesnost sledování trajektorie. Protože trajektorie, která není dopředu známa nelze přesně sledovat, nelze u tohoto bloku přesně definovat trajektorii podobně jako u bloku **MC_MoveDirectAbsolute**. Omezení na rychlost/zrychlení/jerk jsou kontrolována v jednotlivých složkách a celková hodnota ve směru trajektorie může být o něco větší. Zajímavý je režim **CoordSystem** = **TCS**, kde se zadává rychlost v souřadnicích spojených s koncovým efektem robota. Tento režim se velmi hodí pro ruční navádění robota a zcela pochopitelně funguje jen v rychlostním režimu. Navíc se v tomto režimu kontroluje, jestli nedojde k překročení rychlosti motorů (např. při pohybu v blízkosti singulárních bodů) a pokud ano, je pohyb zpomalen.

Tento blok nepropaguje kvalitu signálu. Více informací je uvedeno v sekci [1.4](#).

Vstup

uAxesGroup	Odkaz na skupinu os	Reference
posvel	Vektor požadované polohy nebo rychlosti	Reference
der	Vektor požadované derivace vstupu posvel	Reference
TRACKP	Sledování polohy	Bool

TRACKV	Sledování rychlosti	Bool

Parametr

Velocity	Maximální povolená rychlost [unit/s]	Double (F64)
Acceleration	Maximální povolené zrychlení [unit/s ²]	Double (F64)
Jerk	Maximální povolená změna zrychlení [unit/s ³]	Double (F64)
CoordSystem	Volba souřadného systému	⊙2 Long (I32)
	1 ACS	
	2 MCS	
	3 PCS	
	4 TCS	
LimitMode	Význam limitů rychlosti/zrychlení/změny zrychlení	⊙1 Long (I32)
	1 Relativní (k defaultním hodnotám)	
	2 Absolutní [unit/s unit/s ² ...]	
iLen	Počet vzorků pro odhad rychlosti /zrychlení	↓-1 ↑99 Long (I32)

Výstup

yAxesGroup	Odkaz na skupinu os	Reference
InTrack	Příznak dosažení sledované polohy / rychlosti	Bool
CommandAborted	Příznak přerušení funkce bloku	Bool
Busy	Příznak, že algoritmus ještě neskončil	Bool
Active	Příznak, že blok řídí osu	Bool
Error	Příznak chyby	Bool
ErrorID	Výsledek poslední operace	Error
	i kód chyby REXYGENu	

Kapitola 23

CanDrv – Komunikace po sběrnici CAN

Obsah

CanItem – Další přijatá zpráva sběrnice CAN	908
CanRecv – Přijetí zprávy sběrnice CAN	909
CanSend – Odeslání zprávy na sběrnici CAN	911

Knihovna CanDrv je určena pro zpracování komunikace po sběrnici CAN (Controller Area Network) v systému REXYGEN. Obsahuje [CanItem](#) pro správu datových položek CAN, [CanRecv](#) pro příjem zpráv ze sběrnice a [CanSend](#) pro odesílání zpráv. Tato knihovna poskytuje zásadní nástroje pro efektivní a spolehlivou komunikaci přes síť CAN, usnadňuje výměnu dat a řídicích příkazů mezi různými komponentami systému.

CanItem – Další přijatá zpráva sběrnice CAN

Symbol bloku

Licence: [CANDRV](#)



Popis funkce

Blok **CanItem** se používá ve spojení s blokem **CanRecv**. Vstup **uRef** musí být připojen k výstupu **itemRef** bloku **CanRecv** (buď přímo nebo nepřímo připojením na výstup **yRef** již připojeného bloku).

CanItem zobrazuje starší zprávy sběrnice CAN, které prošly filtrem v připojeném bloku **CanRecv**.

Pokud je k jednomu bloku **CanRecv** připojeno (přímo i nepřímo) více bloků **CanItem**, zprávy jsou zobrazovány podle pořadí vykonávání bloků **CanItem**, takže 1. blok **CanItem** zobrazuje předposlední přijatou zprávu (poslední je přímo v bloku **CanRecv**), 2. blok **CanItem** zobrazuje 3. od konce přijatou zprávu, atd. Proto se doporučuje připojovat vždy následující blok **CanItem** na **yRef** předcházejícího bloku **CanItem** aby bylo jasné pořadí.

Dokud nepříjde dostatek zpráv, blok zobrazuje na výstupech náhradní hodnoty **msgId** = -1 a **length** = -1.

Výstup **DRDY** = on pokud zobrazovaná zpráva (t.j. hodnoty na výstupech **msgId**, **data**, **length**) přišla po sběrnici CAN během poslední periody (t.j. po minulém spuštění bloku). V opačném případě je zpráva starší (tj. už zpracovaná a tedy neplatná), ale ponechává se na výstupech pro snazší kontrolu a ladění aplikace.

Tento blok nepropaguje kvalitu signálu. Více informací je uvedeno v sekci [1.4](#).

Vstup

uRef	Odkaz na další přijaté packety	Reference

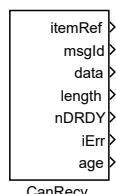
Výstup

yRef	Odkaz na další přijaté packety	Reference
msgId	Číslo zprávy (COB-ID)	Long (I32)
data	Data zprávy (max. 8 bajtů, nejnižší byte první)	Large (I64)
	↓-9.22337E+18 ↑9.22337E+18	
length	Počet datových bajtů zprávy	↓0 ↑8 Long (I32)
DRDY	Přijata zpráva během poslední periody	Bool

CanRecv – Přijetí zprávy sběrnice CAN

Symbol bloku

Licence: [CANDRV](#)



Popis funkce

Blok **CanRecv** slouží k přijetí zprávy na sběrnici CAN. Blok přijímá jen zprávy které odpovídají filtru (parametry **filterId**, **filterIdMask**, **filterLength**, **RTR**, **EXT**).

Počet zpráv, které prošly filtrem za periodu (tj. od minulého spuštění bloku) určuje výstup **nRDY**.

Poslední přijatá zpráva je zobrazena na výstupech **msgId**, **data**, **length**. Starší zprávy (s ohledem na parametr **nmax**) jsou dostupné pomocí bloku **CanItem** navázaného na výstup **itemRef**.

Pro správnou funkci blok musí být napojen na ovladač **CanDrv**, který je v režimu **simpleCAN** (tj. **NodeMode=256**). To se provede pojmenováním bloku dle vzoru **<DRV>__<signal>** (stejně jako u bloků **Goto**, **OUTSTD**, **OUTQAD**, apod.), tj. název bloku musí začínat názvem driveru a dvěma podtržítky následované názvem signálu, přičemž název signálu může být v tomto případě libovolný.

Blok umožňuje přijímat zprávy s krátkým (11bitů) i dlouhým (29 bitů) číslem zprávy (řídí se parametrem **EXT**) a také zprávy pro vyžádání zprávy (parametr **RTR**). FD režim (který umožňuje zprávy s až 64 bajty) není podpořen.

Tento blok nepropaguje kvalitu signálu. Více informací je uvedeno v sekci [1.4](#).

Parametr

filterId	ID zpráv, které mají být přijímány tímto blokem	Long (I32)
	↓0 ↑536870911	
filterIdMask	Maska pro parametr filterId (označuje platné bity)	Long (I32)
	↓0 ↑536870911	
filterLength	Počet byte ve zprávách, které mají být přijímány tímto blokem	Long (I32)
	(-1 povoluje libovolnou délku)	↓-1 ↑8
RTR	Příznak žádost o zprávu (RequestToReceive)	⊙on Bool
EXT	Rozšířený formát čísla zprávy (29bitů)	⊙on Bool
timeout	Pokud během této doby nepřijde packet, je indikována chyba [s]	Double (F64)
		↓0.0

nmax	Maximální počet zpráv přijatých blokem během jedné periody ↓1 ↑255	Long (I32)

Výstup

itemRef	Odkaz na další přijaté packety	Reference
msgId	Číslo zprávy (COB-ID)	Long (I32)
data	Data zprávy (max. 8 bajtů, nejnižší byte první) ↓-9.22337E+18 ↑9.22337E+18	Large (I64)
length	Počet datových bajtů zprávy ↓0 ↑8	Long (I32)
nDRDY	Počet přijatých zpráv ↑255	Word (U16)
iErr	Kód chyby	Error
age	Čas od poslední přijaté zprávy [s] ↓0.0	Double (F64)

CanSend – Odeslání zprávy na sběrnici CAN

Symbol bloku

Licence: [CANDRV](#)



Popis funkce

Blok `CanSend` slouží k odeslání zprávy po sběrnici CAN. Zpráva je určena pomocí vstupů `msgId`, `data`, `length` a parametrů `RTR`, `EXT`. Zpráva se odešle jen pokud je vstup `RUN` nastaven na `on`.

Pro správnou funkci blok musí být napojen na ovladač `CanDrv`, který je v režimu `simpleCAN` (tj. `NodeMode=256`). To se provede pojmenováním bloku dle vzoru `<DRV>__<signal>` (stejně jako u bloků [Goto](#), [OUTSTD](#), [OUTQAD](#), apod.), tj. název bloku musí začínat názvem driveru a dvěma podtržítka následované názvem signálu, přičemž název signálu může být v tomto případě libovolný.

Blok umožňuje posílat zprávy s krátkým (11bitů) i dlouhým (29 bitů) číslem zprávy (řídí se parametrem `EXT`) a také vyžádat si poslání zprávy (parametr `RTR`). FD režim (který umožňuje zprávy s až 64 bajty) není podpořen.

Tento blok nepropaguje kvalitu signálu. Více informací je uvedeno v sekci [1.4](#).

Vstup

<code>msgId</code>	Číslo zprávy (COB-ID)	↓0 ↑536870911	Long (I32)
<code>length</code>	Počet datových bajtů zprávy	↓0 ↑8	Long (I32)
<code>data</code>	Data zprávy (max. 8 bajtů, nejnižší byte první)	↓-9.22337E+18 ↑9.22337E+18	Large (I64)
<code>RUN</code>	Povolení odeslání zprávy		Bool

Parametr

<code>RTR</code>	Příznak žádost o zprávu (RequestToReceive)	⊙on	Bool
<code>EXT</code>	Rozšířený formát čísla zprávy (29bitů)	⊙on	Bool

Výstup

<code>iErr</code>	Kód chyby	Error

Kapitola 24

OpcUaDrv – Komunikace pomocí OPC UA

Obsah

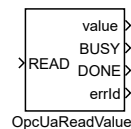
OpcUaReadValue – Čtení hodnoty protokolem OPC UA	914
OpcUaServerValue – Vystavení hodnoty v podobě OPC UA uzlu .	916
OpcUaWriteValue – Zápis hodnoty protokolem OPC UA	918

Knihovna `OpcUaDrv` se specializuje na propojení se servery OPC UA (Open Platform Communications Unified Architecture) pro průmyslovou automatizaci. První blok – [OpcUaReadValue](#) je určen pro čtení dat ze serverů, takže je stěžejní pro sběr dat v automatizovaných systémech. Blok [OpcUaWriteValue](#) umožňuje zápis dat na servery, což umožňuje řízení a provádění příkazů. Blok [OpcUaServerValue](#) navíc usnadňuje monitorování a správu hodnot serveru. Tato knihovna slouží jako kritický nástroj pro bezproblémovou komunikaci a interakci se servery OPC UA, čímž rozšiřuje možnosti automatizačních systémů.

OpcUaReadValue – Čtení hodnoty protokolem OPC UA

Symbol bloku

Licence: [ADVANCED](#)



Popis funkce

Tento funkční blok je závislý na ovladači protokolu OPC UA. Je doporučeno si před použitím přečíst manuál **OpcUaDrv** ovladače [27].

Blok **OpcUaReadValue** slouží pro čtení hodnoty OPC UA uzlu prostřednictvím spojení, které udržuje ovladač **OpcUaDrv** v módu OPC UA Klient.

První dva parametry bloku jsou **NodeId** a **NodeId_type**. **NodeId_type** určuje, jaký typ identifikátoru je očekáván v parametru **NodeId**. Pokud bude vybraný typ jedním z typů **string**, **numeric** nebo **guid**, pak by parametr **NodeId** měl obsahovat identifikátor OPC UA uzlu definovaného na serveru s prefixem indexu jmenného prostoru deklarovaného v konfiguraci ovladače odděleného dvojtečkou (např. **1:myNode**).

Pokud je vybrán typ **cesta**, pak by měl parametr **NodeId** obsahovat cestu k požadovanému uzlu ve stromové struktuře serveru. Každá část cesty se skládá z atributu **BrowseName** uzlu opět s prefixem indexu jmenného prostoru z konfigurace ovladače (např. **/1:myDevice/1:myNode**). Cesta je relativní ke složce *Objects* ve stromové struktuře serveru.

Parametr **type** definuje očekávaný datový typ OPC UA uzlu. Blok konvertuje atribut **value** uzlu na specifikovaný datový typ a nastaví hodnotu na svůj výstup **value** v případě úspěchu a nebo nastaví výstup **errId** na příslušný chybový kód.

Tento blok nepropaguje kvalitu signálu. Více informací je uvedeno v sekci 1.4.

Vstup

READ	Povolení běhu algoritmu	Bool

Parametr

NodeId	OPC UA Node Id	String
NodeId_type	Typ Node Id	⊙1 Long (I32)
	1 string	
	2 numeric	
	3 guid	
	4 cesta	

type	Očekávaný typ příchozích dat	⊙1 Long (I32)
1	Bool	
2	Byte (U8)	
3	Short (I16)	
4	Long (I32)	
5	Word (U16)	
6	DWord (U32)	
7	Float (F32)	
8	Double (F64)	
10	Large (I64)	
12	String	

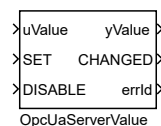
Výstup

value	Výstupní signál	Any
BUSY	Příznak probíhající operace	Bool
DONE	Příznak dokončení transakce	Bool
errId	Kód chyby	Error

OpcUaServerValue – Vystavení hodnoty v podobě OPC UA uzlu

Symbol bloku

Licence: [ADVANCED](#)



Popis funkce

Tento funkční blok je závislý na ovladači protokolu OPC UA. Je doporučeno si před použitím přečíst manuál **OpcUaDrv** ovladače [27].

Blok **OpcUaServerValue** slouží pro vystavení OPC UA uzlu prostřednictvím **OpcUaDrv** ovladače v módu OPC UA Server.

První dva parametry bloku jsou **NodeId** a **NodeId_type**. **NodeId_type** určuje, jaký typ identifikátoru je očekáván v parametru **NodeId**. Parametr **NodeId** určuje identifikátor uzlu, pod kterým bude uzel vystavený na serveru.

Vstupní signál **DISABLE** určuje, zda bude uzel zveřejněn na serveru nebo ne. Pokud je vstupní signál **SET** nastaven na hodnotu **on**, hodnota ze vstupního signálu **uValue** je nastavena atributu **value** OPC UA uzlu. Pokud je hodnota parametru **READONLY** nastavena na **off**, hodnota atributu **value** může být nastavena také prostřednictvím OPC UA protokolu z prostředí mimo algoritmus aplikace.

Výstupní signál **yValue** je nastavena v každém tiku na hodnotu OPC UA uzlu. Parametr **type** určuje datový typ atributu **value** OPC UA uzlu, datový typ vstupu **uValue** i datový typ výstupu **yValue**.

Tento blok nepropaguje kvalitu signálu. Více informací je uvedeno v sekci 1.4.

Vstup

uValue	Vstupní signál	Any
SET	Překopírování hodnoty vstupu do hodnoty OPC UA uzlu	Bool
DISABLE	Deaktivace OPC UA uzlu	Bool

Parametr

NodeId	OPC UA Node Id	String
NodeId_type	Typ id OPC UA uzlu	⊙1 String
	1 string	
	2 numeric	
	3 guid	

type	Datový typ hodnoty	⊙1	Long (I32)
	1 Bool		
	2 Byte (U8)		
	3 Short (I16)		
	4 Long (I32)		
	5 Word (U16)		
	6 DWord (U32)		
	7 Float (F32)		
	8 Double (F64)		
	10 Large (I64)		
	12 String		
BrowseName	Atribut 'browse name' OPC UA uzlu		String
Description	Popiska OPC UA uzlu		String
DisplayName	Zobrazované jméno uzlu OPC UA uzlu		String
READONLY	Nastavení hodnoty OPC uzlu jako pouze pro čtení	⊙on	Bool

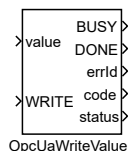
Výstup

yValue	Výstupní signál	Any
CHANGED	Indikace zápisu hodnoty přes OPC UA protocol	Bool
errId	Kód chyby	Error

OpcUaWriteValue – Zápis hodnoty protokolem OPC UA

Symbol bloku

Licence: [ADVANCED](#)



Popis funkce

Tento funkční blok je závislý na ovladači protokolu OPC UA. Je doporučeno si před použitím přečíst manuál **OpcUaDrv** ovladače [27].

Blok **OpcUaWriteValue** slouží pro zápis hodnoty OPC UA uzlu prostřednictvím spojení, které udržuje ovladač **OpcUaDrv** v módu OPC UA Klient.

První dva parametry bloku jsou **NodeId** a **NodeId_type**. **NodeId_type** určuje, jaký typ identifikátoru je očekáván v parametru **NodeId**. Pokud vybraný typ jedním z typů **string**, **numeric** nebo **guid**, pak by parametr **NodeId** měl obsahovat identifikátor OPC UA uzlu definovaného na serveru s prefixem indexu jmenného prostoru deklarovaného v konfiguraci ovladače odděleného dvojtečkou (např. **1:myNode**).

Pokud je vybrán typ **cesta**, pak by měl parametr **NodeId** obsahovat cestu k požadovanému uzlu ve stromové struktuře serveru. Každá část cesty se skládá z atributu **BrowseName** uzlu opět s prefixem indexu jmenného prostoru z konfigurace ovladače (např. **/1:myDevice/1:myNode**). Cesta je relativní ke složce *Objects* ve stromové struktuře serveru.

Parametr **type** definuje očekávaný datový typ OPC UA uzlu. Vstupní signál **value** je převeden na zvolený datový typ a jeho hodnota je poté zapsána na atribut **value** OPC UA uzlu.

Po dokončení operace zápisu výsledný stavový kód operace definovaný standardem OPC UA je zapsán na výstup bloku **code** a jeho textová interpretace je nastavena na výstup bloku **status**.

Tento blok nepropaguje kvalitu signálu. Více informací je uvedeno v sekci 1.4.

Vstup

value	Vstupní signál	Any
WRITE	Povolení běhu algoritmu	Bool

Parametr

NodeId	OPC UA Node Id	String

NodeId_type	Typ Node Id	⊙1	Long (I32)
	1 string		
	2 numeric		
	3 guid		
	4 cesta		
type	Datový typ posílané hodnoty	⊙1	Long (I32)
	1 Bool		
	2 Byte (U8)		
	3 Short (I16)		
	4 Long (I32)		
	5 Word (U16)		
	6 DWord (U32)		
	7 Float (F32)		
	8 Double (F64)		
	10 Large (I64)		
	12 String		

Výstup

BUSY	Příznak probíhající operace	Bool
DONE	Příznak dokončení transakce	Bool
errId	Kód chyby	Error
code	OPC UA výsledný stavový kód operace	DWord (U32)
status	OPC UA stavový kód v textové reprezentaci	String

Kapitola 25

UNIPi – Komunikační bloky pro Unipi

Obsah

IM201CNT – Iris IM201DI, 4 digitální čítače	923
IM201DI – Iris IM201DI, 4 digitální vstupy	924
IM203DO – Iris IM203DO, 8 digitálních výstupů	925
IM203PWM – Iris IM203PWM, 8 PWM výstupů	926
IM204CNT – Iris IM204CNT, 16 digitálních čítačů	927
IM204DI – Iris IM204DI, 16 digitálních vstupů	928
IM205DO – Iris IM205DO, 16 digitálních výstupů	929
IM205PWM – Iris IM205PWM, 16 digitálních výstupů	930
IM301CNT – Iris IM301CNT, 2 digitální čítače	932
IM301DI – Iris IM301DI, 2 digitální vstupy	933
IM301DO – Iris IM301DO, 2 digitální výstupy	934
IM502AO – Iris IM502AO, 4 analogové výstupy	935
IM503AI – Iris IM503AI, 8 analogových vstupů	936
IM504RI – Iris IM504RI, 8 odporových nebo teplotních vstupů	938
IM506AI – Iris IM506AI, 2 analogové vstupy	940
IM506AO – Iris IM506AO, 1 analogový výstup	942
IRIS_MODULE – Iris - Informace z popisu modulu	943
UNIPi_CHANNEL – Informace o Iris modulu nebo Patron sekci	944
UNIPi_PRODUCT – Informace z popisu produktu	946
UNIPi_S1AI – Patron sekce 1, analogový vstup	947
UNIPi_S1AOR – Patron sekce 1, analogový výstup nebo odporový vstup	948
UNIPi_S1CNT – Patron sekce 1, čítače	949
UNIPi_S1DI – Patron sekce 1, digitální vstupy	950

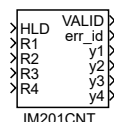
UNIPi_S1DO – Patron sekce 1, digitální výstupy	951
UNIPi_S1LED – Patron sekce 1, LED výstupy	952
UNIPi_S1PWM – Patron sekce 1, PWM výstupy	953
UNIPi_S2AI – Patron sekce 2, analogové vstupy	954
UNIPi_S2AO – Patron sekce 2, analogové výstupy	955
UNIPi_S2CNT – Patron sekce 2, čítače	956
UNIPi_S2DI – Patron sekce 2, digitální vstupy	957
UNIPi_S2RO – Patron sekce 2, reléové výstupy	958

Tato knihovna slouží k ovládání a monitorování zařízení Unipi. Obsahuje bloky pro čtení a zápis digitálních a analogových vstupů a výstupů, bloky pro ovládání relé, PWM výstupů, LED diod a čtení čítačů. Pro čtení sběrnic slouží drivery jako je OwsDrv nebo MbDrv. Bloky fungují v souladu s dokumentací výrobce [1], kde lze najít technické podrobnosti k jednotlivým zařízením a jejich vstupům a výstupům.

IM201CNT – Iris IM201DI, 4 digitální čítače

Symbol bloku

Licence: [UNIPi](#)



Popis funkce

Blok IM201CNT umožňuje využití digitálních vstupů jako čítačů na modulu Unipi Iris. Více informací najdete na webových stránkách výrobce [1].

Tento blok nepropaguje kvalitu signálu. Více informací je uvedeno v sekci 1.4.

Vstup

HLD	Pozastavení	Bool
R1..R4	Resetování čítače	Bool

Parametr

slot	Slot	Long (I32)
0	 Nepoužito	
12	 Slot #12	
22	 Slot #22	
32	 Slot #32	
42	 Slot #42	
52	 Slot #52	
11	 Slot #11	
21	 Slot #21	
debounce1..debounce4	Filtr zákmitu v 100 μ s	↓0 ↑65535 ⊙1 Long (I32)
reset_value1..reset_value4	Hodnota pro restart čítače	↑4294967295 Word (U16)

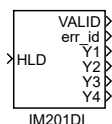
Výstup

VALID	Indikátor validity dat	Bool
err_id	ID chyby	Long (I32)
y1..y4	Hodnota čítače	↑4294967295 Word (U16)

IM201DI – Iris IM201DI, 4 digitální vstupy

Symbol bloku

Licence: [UNIPi](#)



Popis funkce

Blok IM201DI umožňuje čtení digitálních vstupů na modulu Unipi Iris. Více informací najdete na webových stránkách výrobce [1].

Tento blok nepropaguje kvalitu signálu. Více informací je uvedeno v sekci 1.4.

Vstup

HLD	Pozastavení	Bool

Parametr

slot	Slot	Long (I32)
	0 Nepoužito	
	12 Slot #12	
	22 Slot #22	
	32 Slot #32	
	42 Slot #42	
	52 Slot #52	
	11 Slot #11	
	21 Slot #21	
debounce1..debounce4	Filtr zákmitu v 100 μs	↓0 ↑65535 ⊙1 Long (I32)

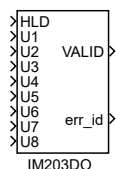
Výstup

VALID	Indikátor validity dat	Bool
err_id	ID chyby	Long (I32)
Y1..Y4	Digitální vstup	Bool

IM203D0 – Iris IM203DO, 8 digitálních výstupů

Symbol bloku

Licence: [UNIPi](#)



Popis funkce

Blok IM203D0 umožňuje ovládání digitálních výstupů na modulu Unipi Iris. Více informací najdete na webových stránkách výrobce [1].

Tento blok nepropaguje kvalitu signálu. Více informací je uvedeno v sekci 1.4.

Vstup

HLD	Pozastavení	Bool
U1..U8	Digitální výstup	Bool

Parametr

slot	Slot	Long (I32)
0	Nepoužito	
12	Slot #12	
22	Slot #22	
32	Slot #32	
42	Slot #42	
52	Slot #52	
11	Slot #11	
21	Slot #21	

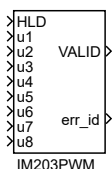
Výstup

VALID	Indikátor validity dat	Bool
err_id	ID chyby	Long (I32)

IM203PWM – Iris IM203PWM, 8 PWM výstupů

Symbol bloku

Licence: [UNIPi](#)



Popis funkce

Blok IM203PWM umožňuje ovládání digitálních výstupů na modulu Unipi Iris v režimu pulsně šířkové modulace (PWM). Více informací lze nalézt na stránkách výrobce [1].

Tento blok nepropaguje kvalitu signálu. Více informací je uvedeno v sekci 1.4.

Vstup

HLD	Pozastavení	Bool
u1..u8	PWM výstup	Word (U16)

Parametr

slot	Slot	Long (I32)
	0 Nepoužito	
	12 Slot #12	
	22 Slot #22	
	32 Slot #32	
	42 Slot #42	
	52 Slot #52	
	11 Slot #11	
	21 Slot #21	
freq	Frekvence	Long (I32)
	0 1kHz	
	1 100Hz	
	2 Vlastní dělička	
prescaler	Prescaler	↓0 ↑65535 Long (I32)

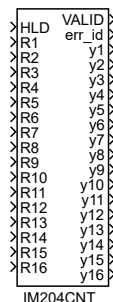
Výstup

VALID	Indikátor validity dat	Bool
err_id	ID chyby	Long (I32)

IM204CNT – Iris IM204CNT, 16 digitálních čítačů

Symbol bloku

Licence: [UNIPi](#)



Popis funkce

Blok IM204CNT umožňuje využití digitálních vstupů jako čítačů na modulu Unipi Iris. Více informací najdete na webových stránkách výrobce [1].

Tento blok nepropaguje kvalitu signálu. Více informací je uvedeno v sekci 1.4.

Vstup

HLD	Pozastavení	Bool
R1..R16	Resetování čítače	Bool

Parametr

slot	Slot	Long (I32)
0	 Nepoužito	
12	 Slot #12	
22	 Slot #22	
32	 Slot #32	
42	 Slot #42	
52	 Slot #52	
11	 Slot #11	
21	 Slot #21	
debounce1..debounce16	Filtr zákmitu v 100 μ s	↓0 ↑65535 ⊙1 Long (I32)
reset_value1..reset_value16	Hodnota pro restart čítače	↑4294967295 Word (U16)

Výstup

VALID	Indikátor validity dat	Bool
err_id	ID chyby	Long (I32)
y1..y16	Hodnota čítače	↑4294967295 Word (U16)

IM204DI – Iris IM204DI, 16 digitálních vstupů

Symbol bloku

Licence: [UNIPÍ](#)



Popis funkce

Blok IM204DI umožňuje čtení digitálních vstupů na modulu Unipi Iris. Více informací najdete na webových stránkách výrobce [1].

Tento blok nepropaguje kvalitu signálu. Více informací je uvedeno v sekci 1.4.

Vstup

HLD	Pozastavení	Bool

Parametr

slot	Slot	Long (I32)
	0 Nepoužito	
	12 Slot#12	
	22 Slot#22	
	32 Slot#32	
	42 Slot#42	
	52 Slot#52	
	11 Slot#11	
	21 Slot#21	
debounce1..debounce16	Filtr zákmitu v 100 μs	↓0 ↑65535 ☉1 Long (I32)

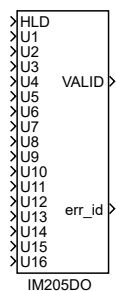
Výstup

VALID	Indikátor validity dat	Bool
err_id	ID chyby	Long (I32)
Y1..Y16	Digitální vstup	Bool

IM205D0 – Iris IM205DO, 16 digitálních výstupů

Symbol bloku

Licence: [UNIPi](#)



Popis funkce

Blok IM205D0 umožňuje ovládání digitálních výstupů na modulu Unipi Iris. Více informací najdete na webových stránkách výrobce [1].

Tento blok nepropaguje kvalitu signálu. Více informací je uvedeno v sekci 1.4.

Vstup

HLD	Pozastavení	Bool
U1..U16	Digitální výstup	Bool

Parametr

slot	Slot	Long (I32)
0	 Nepoužito	
12	 Slot #12	
22	 Slot #22	
32	 Slot #32	
42	 Slot #42	
52	 Slot #52	
11	 Slot #11	
21	 Slot #21	

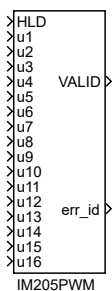
Výstup

VALID	Indikátor validity dat	Bool
err_id	ID chyby	Long (I32)

IM205PWM – Iris IM205PWM, 16 digitálních výstupů

Symbol bloku

Licence: [UNIPÍ](#)



Popis funkce

Blok IM205PWM umožňuje ovládání digitálních výstupů na modulu Unipi Iris v režimu pulsně šířkové modulace (PWM). Více informací lze nalézt na stránkách výrobce [1].

Tento blok nepropaguje kvalitu signálu. Více informací je uvedeno v sekci 1.4.

Vstup

HLD	Pozastavení	Bool
u1..u16	PWM výstup	Word (U16)

Parametr

slot	Slot	Long (I32)
	0 Nepoužito	
	12 Slot#12	
	22 Slot#22	
	32 Slot#32	
	42 Slot#42	
	52 Slot#52	
	11 Slot#11	
	21 Slot#21	
freq	Frekvence	Long (I32)
	0 1kHz	
	1 100Hz	
	2 Vlastní dělička	
prescaler	Prescaler	↓0 ↑65535 Long (I32)

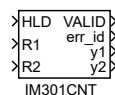
Výstup

VALID	Indikátor validity dat
err_id	ID chyby

Bool
Long (I32)

IM301CNT – Iris IM301CNT, 2 digitální čítače

Symbol bloku

Licence: [UNIPÍ](#)

Popis funkce

Blok IM301CNT umožňuje využití digitálních vstupů jako čítačů na modulu Unipi Iris. Více informací najdete na webových stránkách výrobce [1].

Tento blok nepropaguje kvalitu signálu. Více informací je uvedeno v sekci 1.4.

Vstup

HLD	Pozastavení	Bool
R1	Resetování čítače	Bool
R2	Resetování čítače	Bool

Parametr

slot	Slot	Long (I32)
	0 Nepoužito	
	12 Slot#12	
	22 Slot#22	
	32 Slot#32	
	42 Slot#42	
	52 Slot#52	
	11 Slot#11	
	21 Slot#21	
debounce1	Filtr zákmitu v 100 μ s	↓0 ↑65535 ⊙1 Long (I32)
debounce2	Filtr zákmitu v 100 μ s	↓0 ↑65535 ⊙1 Long (I32)
reset_value1	Hodnota pro restart čítače	↑4294967295 Word (U16)
reset_value2	Hodnota pro restart čítače	↑4294967295 Word (U16)

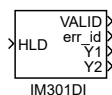
Výstup

VALID	Indikátor validity dat	Bool
err_id	ID chyby	Long (I32)
y1	Hodnota čítače	↑4294967295 Word (U16)
y2	Hodnota čítače	↑4294967295 Word (U16)

IM301DI – Iris IM301DI, 2 digitální vstupy

Symbol bloku

Licence: [UNIPi](#)



Popis funkce

Blok IM301DI umožňuje čtení digitálních vstupů na modulu Unipi Iris. Více informací najdete na webových stránkách výrobce [\[1\]](#).

Tento blok nepropaguje kvalitu signálu. Více informací je uvedeno v sekci [1.4](#).

Vstup

HLD	Pozastavení	Bool

Parametr

slot	Slot	Long (I32)
	0 Nepoužito	
	12 Slot #12	
	22 Slot #22	
	32 Slot #32	
	42 Slot #42	
	52 Slot #52	
	11 Slot #11	
	21 Slot #21	
debounce1	Filtr zákmitu v 100 μ s	↓0 ↑65535 ⊙1 Long (I32)
debounce2	Filtr zákmitu v 100 μ s	↓0 ↑65535 ⊙1 Long (I32)

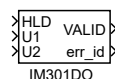
Výstup

VALID	Indikátor validity dat	Bool
err_id	ID chyby	Long (I32)
Y1	Digitální vstup	Bool
Y2	Digitální vstup	Bool

IM301D0 – Iris IM301DO, 2 digitální výstupy

Symbol bloku

Licence: [UNIPi](#)



Popis funkce

Blok IM301D0 umožňuje ovládání digitálních výstupů na modulu Unipi Iris. Více informací najdete na webových stránkách výrobce [1].

Tento blok nepropaguje kvalitu signálu. Více informací je uvedeno v sekci 1.4.

Vstup

HLD	Pozastavení	Bool
U1	Digitální výstup	Bool
U2	Digitální výstup	Bool

Parametr

slot	Slot	Long (I32)
	0	Nepoužito
	12	Slot #12
	22	Slot #22
	32	Slot #32
	42	Slot #42
	52	Slot #52
	11	Slot #11
	21	Slot #21

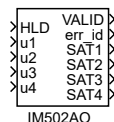
Výstup

VALID	Indikátor validity dat	Bool
err_id	ID chyby	Long (I32)

IM502A0 – Iris IM502A0, 4 analogové výstupy

Symbol bloku

Licence: [UNIFI](#)



Popis funkce

Blok IM502A0 umožňuje ovládání analogových výstupů na modulu Unipi Iris. Více informací najdete na webových stránkách výrobce [\[1\]](#).

Tento blok nepropaguje kvalitu signálu. Více informací je uvedeno v sekci [1.4](#).

Vstup

HLD	Pozastavení	Bool
u1..u4	Analogový výstup	Double (F64)

Parametr

slot	Slot	Long (I32)
0	Nepoužito	
12	Slot #12	
22	Slot #22	
32	Slot #32	
42	Slot #42	
52	Slot #52	
11	Slot #11	
21	Slot #21	

Výstup

VALID	Indikátor validity dat	Bool
err_id	ID chyby	Long (I32)
SAT1..SAT4	Saturace analogového výstupu	Bool

IM503AI – Iris IM503AI, 8 analogových vstupů

Symbol bloku

Licence: [UNIPÍ](#)

Popis funkce

Blok IM503AI umožňuje čtení analogových vstupů na modulu Unipi Iris. Více informací najdete na webových stránkách výrobce [1].

Tento blok nepropaguje kvalitu signálu. Více informací je uvedeno v sekci 1.4.

Vstup

HLD Pozastavení Bool

Parametr

slot	Slot	Long (I32)
	0 Nepoužito	
	12 Slot#12	
	22 Slot#22	
	32 Slot#32	
	42 Slot#42	
	52 Slot#52	
	11 Slot#11	
	21 Slot#21	
mode1..mode8	Režim	⊙1 Long (I32)
	0 Nepoužito	
	1 Napětí	
	2 Proud	
average_mode	Režim průměrování	⊙1 Long (I32)
	0 Uživatelské	
	1 50-60Hz	
	2 50Hz	
	3 60Hz	
average_window	Okno průměrování	↓1 Word (U16)
average_count	Počet průměrování	↓1 ↑256 Word (U16)

Výstup

VALID	Indikátor validity dat
err_id	ID chyby
y1..y8	Analogový vstup

Bool
Long (I32)
Double (F64)

IM504RI – Iris IM504RI, 8 odporových nebo teplotních vstupů

Symbol bloku

Licence: [UNIPÍ](#)



Popis funkce

Blok IM504RI umožňuje čtení odporových nebo teplotních vstupů na modulu Unipi Iris. Více informací najdete na webových stránkách výrobce [1].

Tento blok nepropaguje kvalitu signálu. Více informací je uvedeno v sekci 1.4.

Vstup

HLD	Pozastavení	Bool

Parametr

slot	Slot	Long (I32)
	0 Nepoužito	
	12 Slot#12	
	22 Slot#22	
	32 Slot#32	
	42 Slot#42	
	52 Slot#52	
	11 Slot#11	
	21 Slot#21	
mode1..mode8	Režim	Long (I32)
	0 Nepoužito	
	1 PT100	
	2 PT1000	
	3 Odpor	
average1..average8	Průměrování	Long (I32)
	0 Vypnuto	
	1 Rychlé	
	2 Pomalé	
filter	Číslo chyby	Long (I32)
	0 60HZ	
	1 50HZ	

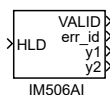
Výstup

VALID	Indikátor validity dat
err_id	ID chyby
y1..y8	Měřená hodnota

Bool
Long (I32)
Double (F64)

IM506AI – Iris IM506AI, 2 analogové vstupy

Symbol bloku

Licence: [UNIPÍ](#)

Popis funkce

Blok IM506AI umožňuje čtení analogových vstupů na modulu Unipi Iris. Více informací najdete na webových stránkách výrobce [1].

Tento blok nepropaguje kvalitu signálu. Více informací je uvedeno v sekci 1.4.

Vstup

HLD	Pozastavení	Bool

Parametr

slot	Slot	Long (I32)
	0 Nepoužito	
	12 Slot #12	
	22 Slot #22	
	32 Slot #32	
	42 Slot #42	
	52 Slot #52	
	11 Slot #11	
	21 Slot #21	
mode1	Režim	Long (I32)
	0 Nepoužito	
	1 Napětí	
	2 Proud	
mode2	Režim	Long (I32)
	0 Nepoužito	
	1 Napětí	
	2 Proud	
average_mode	Režim průměrování	⊙1 Long (I32)
	0 Uživatelské	
	1 50-60Hz	
	2 50Hz	
	3 60Hz	
average_window	Okno průměrování	↓1 Word (U16)
average_count	Počet průměrování	↓1 ↑256 Word (U16)

Výstup

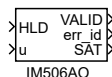
VALID	Indikátor validity dat
err_id	ID chyby
y1	Analogový vstup
y2	Analogový vstup

Bool
Long (I32)
Double (F64)
Double (F64)

IM506AO – Iris IM506AO, 1 analogový výstup

Symbol bloku

Licence: [UNIPi](#)



Popis funkce

Blok IM506AO umožňuje ovládání analogových výstupů na modulu Unipi Iris. Více informací najdete na webových stránkách výrobce [1].

Tento blok nepropaguje kvalitu signálu. Více informací je uvedeno v sekci 1.4.

Vstup

HLD	Pozastavení	Bool
u	Analogový výstup	Double (F64)

Parametr

slot	Slot	Long (I32)
	0 Nepoužito	
	12 Slot#12	
	22 Slot#22	
	32 Slot#32	
	42 Slot#42	
	52 Slot#52	
	11 Slot#11	
	21 Slot#21	

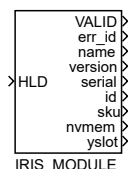
Výstup

VALID	Indikátor validity dat	Bool
err_id	ID chyby	Long (I32)
SAT	Saturace analogového výstupu	Bool

IRIS_MODULE – Iris - Informace z popisu modulu

Symbol bloku

Licence: [UNIPi](#)



Popis funkce

Blok `IRIS_MODULE` načítá informace o modulu Unipi Iris a zobrazuje je ve svých výstupech. Další informace najdete na stránkách výrobce [\[1\]](#).

Tento blok nepropaguje kvalitu signálu. Více informací je uvedeno v sekci [1.4](#).

Vstup

HLD	Pozastavení	Bool

Parametr

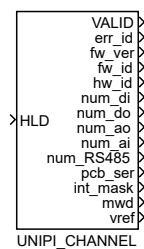
slot	Slot	Long (I32)
	0 Nepoužito	
	12 Slot #12	
	22 Slot #22	
	32 Slot #32	
	42 Slot #42	
	52 Slot #52	
	11 Slot #11	
	21 Slot #21	

Výstup

VALID	Indikátor validity dat	Bool
err_id	ID chyby	Long (I32)
name	Název modulu	String
version	Verze modulu	String
serial	Sériové číslo modulu	String
id	ID modulu	String
sku	SKU	String
nvmem	Nvmem	String
yslot	Slot	String

UNIPÍ_CHANNEL – Informace o Iris modulu nebo Patron sekci

Symbol bloku

Licence: [UNIPÍ](#)

Popis funkce

Blok UNIPÍ_CHANNEL načítá informace o modulu Unipi Iris nebo sekci Unipi Patron a zobrazuje je ve svých výstupech. Další informace najdete na stránkách výrobce [\[1\]](#).

Tento blok nepropaguje kvalitu signálu. Více informací je uvedeno v sekci [1.4](#).

Vstup

HLD	Pozastavení	Bool

Parametr

slot	Slot	Long (I32)
	0 Nepoužito	
	1 Sekce#1	
	2 Sekce#2	
	3 Sekce#3	
	12 Slot#12	
	22 Slot#22	
	32 Slot#32	
	42 Slot#42	
	52 Slot#52	
	11 Slot#11	
	21 Slot#21	

Výstup

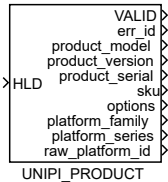
VALID	Indikátor validity dat	Bool
err_id	ID chyby	Long (I32)
fw_ver	Verze Firmware	Word (U16)
fw_id	ID Firmware	Word (U16)
hw_id	ID Hardware	Word (U16)

num_di	Počet digitálních vstupů	Word (U16)
num_do	Počet digitálních výstupů	Word (U16)
num_ao	Počet analogových výstupů	Word (U16)
num_ai	Počet analogových vstupů	Word (U16)
num_RS485	Počet interních RS485 linek	Word (U16)
pcb_ser	Sériové číslo DPS	DWord (U32)
int_mask	Maska přerušení	Word (U16)
mwd	Section MasterWatchDog (MWD) Timeout (in 1ms)	Word (U16)
vref	VRef of MCU	Word (U16)

UNIFI_PRODUCT – Informace z popisu produktu

Symbol bloku

Licence: [UNIFI](#)



Popis funkce

Blok UNIFI_PRODUCT načítá informace o produktu Unipi a zobrazuje je ve svých výstupech. Další informace najdete na stránkách výrobce [\[1\]](#).

Tento blok nepropaguje kvalitu signálu. Více informací je uvedeno v sekci [1.4](#).

Vstup

HLD	Pozastavení	Bool

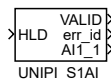
Výstup

VALID	Indikátor validity dat	Bool
err_id	ID chyby	Long (I32)
product_model	Model produktu	String
product_version	Verze produktu	String
product_serial	Sériové číslo produktu	String
sku	SKU	String
options	Motnosti	String
platform_family	Rodina platformy	String
platform_series	Řada platformy	String
raw_platform_id	RAW ID platformy	String

UNIPi_S1AI – Patron sekce 1, analogový vstup

Symbol bloku

Licence: [UNIPi](#)



Popis funkce

Blok **UNIPi_S1AI** umožňuje čtení hodnot analogového vstupu umístěného v sekci 1 zařízení Unipi Patron. Tuto sekci obsahují všechny modely Unipi Patron. Více informací lze nalézt na stránkách výrobce [1].

Tento blok nepropaguje kvalitu signálu. Více informací je uvedeno v sekci 1.4.

Vstup

HLD	Pozastavení	Bool

Parametr

mode	Režim napětí/proud	⊙1 Long (I32)
	0 Napětí 0 - 10V	
	1 Proud 0 - 20mA	

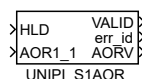
Výstup

VALID	Indikátor validity dat	Bool
err_id	ID chyby	Long (I32)
AI1_1	Analogový vstup	Double (F64)

UNIPi_S1AOR – Patron sekce 1, analogový výstup nebo odporový vstup

Symbol bloku

Licence: [UNIPi](#)



Popis funkce

Blok **UNIPi_S1AOR** umožňuje ovládat analogový výstup umístěný v sekci 1 zařízení Unipi Patron. Tuto sekci obsahují všechny modely Unipi Patron. V závislosti na hodnotě parametru **mode** je možné generovat napětí, proud nebo měřit odpor. Více informací lze nalézt na stránkách výrobce [1].

Tento blok nepropaguje kvalitu signálu. Více informací je uvedeno v sekci 1.4.

Vstup

HLD	Pozastavení	Bool
AOR1_1	Analogový výstup	Bool

Parametr

mode	Režim	⊙1	Long (I32)
	0	Napěťový výstup 0 - 10V	
	1	Proudový výstup 0 - 20mA	
	3	Měření odporu 0 - 2 kΩ	

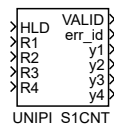
Výstup

VALID	Indikátor validity dat	Bool
err_id	ID chyby	Long (I32)
AORV	Měřený odpor	Double (F64)

UNIP1_S1CNT – Patron sekce 1, čítače

Symbol bloku

Licence: [UNIP1](#)



Popis funkce

Blok UNIP1_S1CNT umožňuje použití digitálních vstupů umístěných v sekci 1 zařízení Unipi Patron v režimu čítačových vstupů. Sekci 1 obsahují všechny modely Unipi Patron. Více informací lze nalézt na stránkách výrobce [1].

Tento blok nepropaguje kvalitu signálu. Více informací je uvedeno v sekci 1.4.

Vstup

HLD	Pozastavení	Bool
R1..R4	Resetování čítače	Bool

Parametr

debounce1..debounce4	Filtr zákmitu v 100 μ s	↓0 ↑65535 ⊙-1	Long (I32)
reset_value1..reset_value4	Hodnota pro restart čítače	↑4294967295	Word (U16)

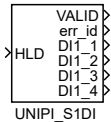
Výstup

VALID	Indikátor validity dat	Bool
err_id	ID chyby	Long (I32)
y1..y4	Hodnota čítače	↑4294967295 Word (U16)

UNIPi_S1DI – Patron sekce 1, digitální vstupy

Symbol bloku

Licence: [UNIPi](#)



Popis funkce

Blok UNIPi_S1DI umožňuje čtení digitálních vstupů umístěných v sekci 1 zařízení Unipi Patron. Tuto sekci obsahují všechny modely Unipi Patron. Více informací lze nalézt na stránkách výrobce [1].

Tento blok nepropaguje kvalitu signálu. Více informací je uvedeno v sekci 1.4.

Vstup

HLD	Pozastavení	Bool

Parametr

debounce1..debounce4	Filtr zákmitu v 100 μs	↓0 ↑65535 ⊖-1	Long (I32)

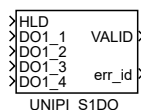
Výstup

VALID	Indikátor validity dat	Bool
err_id	ID chyby	Long (I32)
DI1_1	Digitální vstup	Bool
DI1_2	Digitální vstup	Bool
DI1_3	Digitální vstup	Bool
DI1_4	Digitální vstup	Bool

UNIPIS1D0 – Patron sekce 1, digitální výstupy

Symbol bloku

Licence: [UNIPIS](#)



Popis funkce

Blok UNIPIS1D0 umožňuje použití digitálních výstupů umístěných v sekci 1 zařízení Unipi Patron. Tuto sekci obsahují všechny modely Unipi Patron. Více informací lze nalézt na stránkách výrobce [1].

Tento blok nepropaguje kvalitu signálu. Více informací je uvedeno v sekci 1.4.

Vstup

HLD	Pozastavení	Bool
D01_1	Digitální výstup	Bool
D01_2	Digitální výstup	Bool
D01_3	Digitální výstup	Bool
D01_4	Digitální výstup	Bool

Parametr

enable_DS1..enable_DS4	Povolení přímého spínání	on	Bool
invert_DS1..invert_DS4	Invertovat přímé spínání	on	Bool
toggle_DS1..toggle_DS4	Přepnout přímé spínání	on	Bool

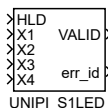
Výstup

VALID	Indikátor validity dat	Bool
err_id	ID chyby	Long (I32)

UNIPi_S1LED – Patron sekce 1, LED výstupy

Symbol bloku

Licence: [UNIPi](#)



Popis funkce

Blok **UNIPi_S1LED** je blok, který slouží k ovládání LED diod umístěných v sekci 1 zařízení Unipi Patron. Tuto sekci obsahují všechny modely Unipi Patron. Více informací lze nalézt na stránkách výrobce [\[1\]](#).

Tento blok nepropaguje kvalitu signálu. Více informací je uvedeno v sekci [1.4](#).

Vstup

HLD	Pozastavení	Bool
X1..X4	LED indikátor	Bool

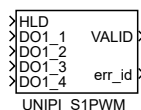
Výstup

VALID	Indikátor validity dat	Bool
err_id	ID chyby	Long (I32)

UNIP1_S1PWM – Patron sekce 1, PWM výstupy

Symbol bloku

Licence: [UNIP1](#)



Popis funkce

Blok UNIP1_S1PWM je blok, který slouží k ovládání digitálních výstupů umístěných v sekci 1 zařízení Unipi Patron v režimu pulsně šířkové modulace (PWM). Sekci 1 obsahují všechny modely Unipi Patron. Více informací lze nalézt na stránkách výrobce [\[1\]](#).

Tento blok nepropaguje kvalitu signálu. Více informací je uvedeno v sekci [1.4](#).

Vstup

HLD	Pozastavení	Bool
D01_1	PWM výstup	Word (U16)
D01_2	PWM výstup	Word (U16)
D01_3	PWM výstup	Word (U16)
D01_4	PWM výstup	Word (U16)

Parametr

prescaler	Prescaler	↓0 ↑65535	Long (I32)
cycle_length	Délka cyklu	↓0 ↑65535	Long (I32)

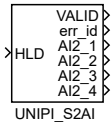
Výstup

err_id	ID chyby	Long (I32)
VALID	Indikátor validity dat	Bool

UNIPi_S2AI – Patron sekce 2, analogové vstupy

Symbol bloku

Licence: [UNIPi](#)



Popis funkce

Blok UNIPi_S2AI umožňuje čtení hodnot analogového vstupu umístěného v sekci 2 zařízení Unipi Patron. Blok je možné použít pouze u konkrétních modelů Patron, které mají více než jednu sekci. Více informací lze nalézt na stránkách výrobce [\[1\]](#).

Tento blok nepropaguje kvalitu signálu. Více informací je uvedeno v sekci [1.4](#).

Vstup

HLD	Pozastavení	Bool

Parametr

mode1..mode4	Režim napětí/proud	⊙1	Long (I32)
0	 Napětí 0 - 10V		
1	 Proud 0 - 20mA		

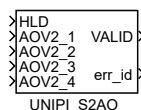
Výstup

VALID	Indikátor validity dat	Bool
err_id	ID chyby	Long (I32)
AI2_1	Analogový vstup	Double (F64)
AI2_2	Analogový vstup	Double (F64)
AI2_3	Analogový vstup	Double (F64)
AI2_4	Analogový vstup	Double (F64)

UNIPi_S2A0 – Patron sekce 2, analogové výstupy

Symbol bloku

Licence: [UNIPi](#)



Popis funkce

Blok **UNIPi_S2A0** umožňuje ovládat analogový výstup umístěný v sekci 2 zařízení Unipi Patron. Blok je možné použít pouze u konkrétních modelů Patron, které mají více než jednu sekci. Více informací lze nalézt na stránkách výrobce [\[1\]](#).

Tento blok nepropaguje kvalitu signálu. Více informací je uvedeno v sekci [1.4](#).

Vstup

HLD	Pozastavení	Bool
AOV2_1	Analogový výstup	Double (F64)
AOV2_2	Analogový výstup	Double (F64)
AOV2_3	Analogový výstup	Double (F64)
AOV2_4	Analogový výstup	Double (F64)

Výstup

VALID	Indikátor validity dat	Bool
err_id	ID chyby	Long (I32)

UNIPi_S2CNT – Patron sekce 2, čítače

Symbol bloku

Licence: [UNIPi](#)

Popis funkce

Blok UNIPi_S2CNT umožňuje použití digitálních vstupů umístěných v sekci 2 zařízení Unipi Patron v režimu čítačových vstupů. Blok je možné použít pouze u konkrétních modelů Patron, které mají více než jednu sekci. Více informací lze nalézt na stránkách výrobce [1].

Tento blok nepropaguje kvalitu signálu. Více informací je uvedeno v sekci 1.4.

Vstup

HLD	Pozastavení	Bool
R1..R4	Resetování čítače	Bool

Parametr

debounce1..debounce4	Filtr zákmitu v 100 μ s	↓0 ↑65535 ⊖-1	Long (I32)
reset_value1..reset_value4	Hodnota pro restart čítače	↑4294967295 ⊕1	Word (U16)

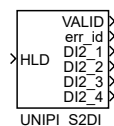
Výstup

VALID	Indikátor validity dat	Bool
err_id	ID chyby	Long (I32)
y1..y4	Hodnota čítače	↑4294967295 Word (U16)

UNIPi_S2DI – Patron sekce 2, digitální vstupy

Symbol bloku

Licence: [UNIPi](#)



Popis funkce

Blok UNIPi_S2DI umožňuje čtení digitálních vstupů umístěných v sekci 2 zařízení Unipi Patron. Blok je možné použít pouze u konkrétních modelů Patron, které mají více než jednu sekci. Více informací lze nalézt na stránkách výrobce [1].

Tento blok nepropaguje kvalitu signálu. Více informací je uvedeno v sekci 1.4.

Vstup

HLD	Pozastavení	Bool

Parametr

debounce1..debounce4	Filtr zákmitu v 100 μ s	↓0 ↑65535 ⊖-1	Long (I32)

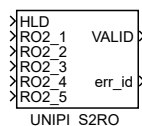
Výstup

VALID	Indikátor validity dat	Bool
err_id	ID chyby	Long (I32)
DI2_1	Digitální vstup	Bool
DI2_2	Digitální vstup	Bool
DI2_3	Digitální vstup	Bool
DI2_4	Digitální vstup	Bool

UNIPi_S2R0 – Patron sekce 2, reléové výstupy

Symbol bloku

Licence: [UNIPi](#)



Popis funkce

Blok UNIPi_S2R0 umožňuje použití reléových výstupů umístěných v sekci 2 zařízení Unipi Patron. Blok je možné použít pouze u konkrétních modelů Patron, které mají více než jednu sekci. Více informací lze nalézt na stránkách výrobce [1].

Tento blok nepropaguje kvalitu signálu. Více informací je uvedeno v sekci 1.4.

Vstup

HLD	Pozastavení	Bool
R02_1	Digitální výstup	Bool
R02_2	Digitální výstup	Bool
R02_3	Digitální výstup	Bool
R02_4	Digitální výstup	Bool
R02_5	Digitální výstup	Bool

Parametr

enable_DS1..enable_DS4	Povolení přímého spínání	⊙on	Bool
invert_DS1..invert_DS4	Invertovat přímé spínání	⊙on	Bool
toggle_DS1..toggle_DS4	Přepnout přímé spínání	⊙on	Bool

Výstup

VALID	Indikátor validity dat	Bool
err_id	ID chyby	Long (I32)

Příloha A

Licenční politika

S uvolněním nové verze 3 systému REXYGEN se firma REX Controls s.r.o. rozhodla zavést i nový systém licencování. Od licencí běhového prostředí systému REXYGEN, které byly vázány na konkrétní cílový hardware podle výkonu tohoto hardwaru přecházíme na spravedlivější systém licencování podle rozsáhlosti (složitosti) cílové aplikace. Od této verze nezáleží na tom, zda aplikace bude provozována na malém a levném počítači (např. Raspberry Pi), podporované rodině PLC, průmyslovém PC nebo např. výkonném serveru ani nezáleží na rodině procesorů nebo operačním systému (Linux, Windows, bez operačního systému).

Flexibilní licencování

Nemusíte se obávat nákladů na novou licenci při rozšíření projektu. Díky novému systému licencování je možné provádět upgrady mezi jednotlivými licenčními úrovněmi, přičemž cena upgradu odpovídá rozdílu cen jednotlivých licencí. Tímto způsobem je umožněno průběžné upgradování bez dalších poplatků.

Licenční skupiny

Po instalaci RexCore na vašem cílovém zařízení musíte na zařízení aktivovat licenci. Pro instrukce si přečtete Příručku k Licencování [28]. Funkčnost runtime můžete ověřit pomocí bezplatné licence „DEMO“. Licence „DEMO“ umožňuje provoz systému REXYGEN po dobu až 2 hodin.

Pro trvalý běh systému REXYGEN na cílovém hardwaru je zapotřebí si zakoupit trvalou licenci. Z hlediska licencování existuje několik verzí runtime modulu RexCore, které poskytují maximální flexibilitu pro jednotlivé projekty. Pro cenu licence je rozhodující velikost a složitost vyvinuté aplikace, kterou určují následující faktory použité v aplikaci (konkrétní projekt systému REXYGEN):

- Funkční bloky
- Ovladače vstupů a výstupů
- Rozsáhlost aplikace

- Rozsáhlost vizualizace a operátorského ovládání (Human-Machine Interface, HMI)
- Další nástroje

Uvedené body jsou podrobněji vysvětleny v následujících odstavcích. Podrobný popis jednotlivých bloků, ovladačů, tvorby HMI a dalších nástrojů je uveden v **uživatelské dokumentaci systému REXYGEN**. V průběhu kompilace projektu jsou všechny relevantní faktory zobrazeny v kompilačním okně, včetně seznamu potřebných licencí pro trvalý provoz aplikace.

Funkční bloky

Rozsáhlá knihovna funkčních bloků je rozdělena do několika licenčních skupin:

- STANDARD – běžné (většinou jednoduché) funkční bloky dostupné ve všech variantách licencí,
- ADVANCED – pokročilé (většinou složitější) funkční bloky dostupné ve všech variantách licencí kromě licence Starter,
- REXLANG – volně programovatelný blok v jazyku velmi podobném jazyku C,
- PYTHON – volně programovatelný blok v jazyku PYTHON.

Další skupiny funkčních bloků mají samostatné licence a fungují jako volitelné knihovny, které lze samostatně dokoupit k licencím Profi S, M a L:

- **AUTOTUNING** – regulátory s automatickým nastavováním parametrů,
- **MOTION CONTROL** – řízení pohybu v jedné ose a více osách,
- **COORDINATED MOTION** – koordinované řízení pohybu (např. robotů),
- **MATRIX** – pokročilé maticové funkční bloky založené na ortogonálních transformacích,
- **OPTIM** – optimalizační algoritmy a solvery.

Dostupnost jednotlivých skupin funkčních bloků je závislá na zvolené licenci. V Tabulce [A.1](#) jsou uvedeny skupiny funkčních bloků a jejich dostupnost v licenčních variantách.

Ovladače vstupů a výstupů

Podobně jako funkční bloky, tak i ovladače jsou rozděleny do několika licenčních skupin:

- Basic – základní ovladače pro vstupy a výstupy nativní pro daný hardware, ovladače přímých vstupů a výstupů (pokud je daný HW má) a ovladač IWire pomocí OWFS,
- IIoT – ovladače pro průmyslový internet věcí (IIoT): MQTT, OPC UA driver, Database access (DbDrv, PqDrv),

Licenční skupina	Starter	Plus			Profi			Ultimate
		S	M	L	S	M	L	
STANDARD	✓	✓	✓	✓	✓	✓	✓	✓
ADVANCED		✓	✓	✓	✓	✓	✓	✓
REXLANG		✓	✓	✓	✓	✓	✓	✓
PYTHON					✓	✓	✓	✓
AUTOTUNING					+	+	+	✓
MOTION CONTROL					+	+	+	✓
COORDINATED MOTION					+	+	+	✓
MATRIX					+	+	+	✓
OPTIM					+	+	+	✓

Tabulka A.1: Funkční bloky (✓ = součástí; + = volitelné)

- Fieldbus – Modbus RTU/TCP (master i slave), CAN/CANopen,
- Ethernet – EtherCAT, Siemens S7.

V Tabulce A.2 jsou uvedeny skupiny ovladačů a jejich dostupnost v licenčních variantách.

Licenční skupina	Starter	Plus			Profi			Ultimate
		S	M	L	S	M	L	
Basic	2	2	2	4	∞	∞	∞	∞
IIoT		1	2	2	4	4	8	∞
Fieldbus			1	2	4	8	12	∞
Ethernet				1	2	4	8	∞

Tabulka A.2: Ovladače vstupů a výstupů (∞ = neomezeno)

Rozsáhlost projektu

Každá varianta licence má omezení na maximální počet bloků a vstupů/výstupů, které může projekt obsahovat. Vyšší licenční skupiny také podporují běh na více jádrech procesoru. Informace jsou přehledně uvedeny v Tabulce A.3.

Licenční skupina	Starter	Plus			Profi			Ultimate
		S	M	L	S	M	L	
Počet bloků	512	1024	2048	4096	6144	10240	12288	∞
Počet vstupů a výstupů	64	128	256	512	1024	2048	4096	∞
Více jader				✓	✓	✓	✓	✓

Tabulka A.3: Rozsáhlost projektu (✓ = součástí; ∞ = neomezeno)

Rozsah vizualizace a operátorského ovládání (Human-Machine Interface, HMI)

Ve verzi 3 systému REXYGEN jsou rozsáhlé vizualizace zpoplatněny. Rozsah je určen počtem tzv. tagů. Vizualizace s až 128 tagy jsou zdarma. Tagy se počítají takto:

- Každá použitá hodnota (vstup nebo displej) představuje 1 tag.
- Každý alokovaný prvek pole je 1 tag. Např. pole alokované na 100 prvků s rozměry 3x4 se počítá za 100 tagů. Pokud takové pole bude mít trvale rozměry 3x4, stačí aby bylo alokováno na 12 prvků.
- Každý signál zobrazovaný v trendu se počítá jako 10 tagů.

Pokud počet tagů ve vizualizaci překročí limit licence, je to viditelně indikováno, ale vizualizace pokračuje v činnosti. **Zakoupením** potřebné licence tato indikace zmizí.

Další nástroje

Systém REXYGEN obsahuje i další nástroje, které jsou pro použití v malém měřítku zdarma. Pro rozsáhlé použití jsou zpoplatněny a je nutné zakoupit samostatnou licenci:

- Výkonný **OPC UA Server**
- **RexArcView** - Prohlížeč dat z archivu

Možnost použití je stejně jako u HMI dána počtem tagů. Pro OPC UA Server se tagy počítají stejně jako u HMI. Pro RexArcView se každý zobrazený signál počítá jako 10 tagů. Počet tagů zdarma pro každý nástroj a verzi licence je uveden v Tabulce A.4.

Licenční skupina	Starter	Plus			Profi			Ultimate
		S	M	L	S	M	L	
HMI	128	128	128	128	128	128	128	128
OPC UA	32	32	32	32	512	512	512	512
RexArcView	64	64	64	64	64	64	64	64

Tabulka A.4: Počet tagů zdarma (více tagů lze zakoupit samostatně)

Podrobné informace o licencování jednotlivých funkčních bloků jsou uvedeny v příloze B.

Příloha B

Seznam funkčních bloků a jejich licencování

Aby bylo dosaženo maximální flexibility pro různé projekty, jsou funkční bloky systému REXYGEN licencovány po skupinách. Tabulka níže ukazuje příslušnost funkčních bloků k jednotlivým licenčním skupinám. Bližší informace o možnostech licencování jsou uvedeny v příloze [A](#).

Jméno bloku	Potřebná licence	
	STANDARD	Jiná
ABS	•	
ABSR0T		ADVANCED
ACD	•	
ACLEAR	•	
ADD	•	
ADDHEXD	•	
ADDOCT	•	
ADDQUAD	•	
AFLUSH	•	
ALARMS	•	
ALB	•	
ALBI	•	
ALM	•	
ALMI	•	
ALN	•	
ALNI	•	
AND	•	
ANDHEXD	•	
ANDOCT	•	
ANDQUAD	•	

Seznam pokračuje na další stránce...

Jméno bloku	Potřebná licence	
	STANDARD	Jiná
ANLS	•	
ARC	•	
ARLY	•	
ARS	•	
ASW		ADVANCED
ATMT	•	
AVG	•	
AVS		ADVANCED
AVSI		ADVANCED
BDHEXD	•	
BDOCT	•	
BINS	•	
BIS	•	
BISR	•	
BITOP	•	
BMHEXD	•	
BMOCT	•	
BPF	•	
BSFIFO		ADVANCED
BSGET		ADVANCED
BSGETOCT		ADVANCED
BSGETOCTV		ADVANCED
BSGETV		ADVANCED
BSSET		ADVANCED
BSSETOCT		ADVANCED
BSSETOCTV		ADVANCED
BSSETV		ADVANCED
CDELSSM		ADVANCED
CMP	•	
CNA	•	
CNB	•	
CNDR	•	
CNDT	•	
CNE	•	
CNI	•	
CNR	•	
CNS	•	
CNT	•	
CONCAT	•	

Seznam pokračuje na další stránce...

Jméno bloku	Potřebná licence	
	STANDARD	Jiná
CONCAT_DT	•	
COUNT	•	
CSSM		ADVANCED
CanItem		CANDRV
CanRecv		CANDRV
CanSend		CANDRV
DATA	•	
DATE	•	
DATETIME	•	
DDELSSM		ADVANCED
DEL	•	
DELM	•	
DER	•	
DFIR		ADVANCED
DIF	•	
DIV	•	
DSSM		ADVANCED
DT2STR	•	
Display	•	
EAS	•	
EATMT		ADVANCED
EDGE	•	
EKF		MODEL
EMD	•	
EPC		ADVANCED
EQ	•	
EVAR	•	
EXEC	•	
FFT		ADVANCED
FIND	•	
FIWR		ADVANCED
FLCU		ADVANCED
FNX	•	
FNXY	•	
FOPDT	•	
FRID		ADVANCED
From	•	
FromFile	•	
FromWorkspace	•	

Seznam pokračuje na další stránce...

Jméno bloku	Potřebná licence	
	STANDARD	Jiná
GAIN	•	
GETPA	•	
GETPB	•	
GETPI	•	
GETPR	•	
GETPS	•	
GETPX	•	
GRADS		ADVANCED
Goto	•	
GotoTagVisibility	•	
HID		MODEL
HMI	•	
HTTP		ADVANCED
HTTP2		ADVANCED
I3PM		ADVANCED
IADD	•	
IDIV	•	
IM201CNT		UNIPi
IM201DI		UNIPi
IM203D0		UNIPi
IM203PWM		UNIPi
IM204CNT		UNIPi
IM204DI		UNIPi
IM205D0		UNIPi
IM205PWM		UNIPi
IM301CNT		UNIPi
IM301DI		UNIPi
IM301D0		UNIPi
IM502A0		UNIPi
IM503AI		UNIPi
IM504RI		UNIPi
IM506AI		UNIPi
IM506A0		UNIPi
IMOD	•	
IMUL	•	
INCONN	•	
INFO	•	
INHEXD	•	
INOCT	•	

Seznam pokračuje na další stránce...

Jméno bloku	Potřebná licence	
	STANDARD	Jiná
INQUAD	•	
INSTD	•	
INTE	•	
INTSM	•	
IOASYNC	•	
IODRV	•	
IOTASK	•	
IPEN2		MODEL
IPEN2p		MODEL
IPEN2pu		MODEL
IPEN3		MODEL
IPEN3p		MODEL
IPEN3pu		MODEL
IRIS_MODULE		UNIPI
ISSW	•	
ISUB	•	
ITOI	•	
ITOS	•	
Inport	•	
KDER		ADVANCED
LC	•	
LEN	•	
LIN	•	
LLC	•	
LPBRK	•	
LPF	•	
LPI		ADVANCED
LUA		REXLANG
LUAHEXD		REXLANG
LUAOCT		REXLANG
LUAQUAD		REXLANG
MB_DASUM	•	
MB_DAXPY	•	
MB_DCOPY	•	
MB_DDOT	•	
MB_DGEMM	•	
MB_DGEMV	•	
MB_DGER	•	
MB_DNRM2	•	

Seznam pokračuje na další stránce...

Jméno bloku	Potřebná licence	
	STANDARD	Jiná
MB_DROT	•	
MB_DSCAL	•	
MB_DSWAP	•	
MB_DTRMM	•	
MB_DTRMV	•	
MB_DTRSV	•	
MCP_AccelerationProfile		MOTION CONTROL
MCP_CamIn		MOTION CONTROL
MCP_CamTableSelect		MOTION CONTROL
MCP_CombineAxes		MOTION CONTROL
MCP_GearIn		MOTION CONTROL
MCP_GearInPos		MOTION CONTROL
MCP_GroupHalt		COORDINATED MOTION
MCP_GroupInterrupt		COORDINATED MOTION
MCP_GroupSetOverride		COORDINATED MOTION
MCP_GroupSetPosition		COORDINATED MOTION
MCP_GroupStop		COORDINATED MOTION
MCP_Halt		MOTION CONTROL
MCP_HaltSuperimposed		MOTION CONTROL
MCP_Home		MOTION CONTROL
MCP_MoveAbsolute		MOTION CONTROL
MCP_MoveAdditive		MOTION CONTROL
MCP_MoveCircularAbsolute		COORDINATED MOTION
MCP_MoveCircularRelative		COORDINATED MOTION
MCP_MoveContinuousAbsolute		MOTION CONTROL
MCP_MoveContinuousRelative		MOTION CONTROL
MCP_MoveDirectAbsolute		COORDINATED MOTION
MCP_MoveDirectRelative		COORDINATED MOTION
MCP_MoveLinearAbsolute		COORDINATED MOTION
MCP_MoveLinearRelative		COORDINATED MOTION
MCP_MovePath		COORDINATED MOTION
MCP_MovePath_PH		COORDINATED MOTION
MCP_MoveRelative		MOTION CONTROL
MCP_MoveSuperimposed		MOTION CONTROL
MCP_MoveVelocity		MOTION CONTROL
MCP_PhasingAbsolute		MOTION CONTROL
MCP_PhasingRelative		MOTION CONTROL
MCP_PositionProfile		MOTION CONTROL
MCP_SetCartesianTransform		COORDINATED MOTION

Seznam pokračuje na další stránce...

Jméno bloku	Potřebná licence	
	STANDARD	Jiná
MCP_SetKinTransform_Arm		COORDINATED MOTION
MCP_SetKinTransform_Lin		COORDINATED MOTION
MCP_SetKinTransform_Schunk		COORDINATED MOTION
MCP_SetKinTransform_UR		COORDINATED MOTION
MCP_SetOverride		MOTION CONTROL
MCP_Stop		MOTION CONTROL
MCP_TorqueControl		MOTION CONTROL
MCP_VelocityProfile		MOTION CONTROL
MCU	•	
MC_AccelerationProfile		MOTION CONTROL
MC_AddAxisToGroup		COORDINATED MOTION
MC_CamIn		MOTION CONTROL
MC_CamOut		MOTION CONTROL
MC_CombineAxes		MOTION CONTROL
MC_GearIn		MOTION CONTROL
MC_GearInPos		MOTION CONTROL
MC_GearOut		MOTION CONTROL
MC_GroupContinue		COORDINATED MOTION
MC_GroupDisable		COORDINATED MOTION
MC_GroupEnable		COORDINATED MOTION
MC_GroupHalt		COORDINATED MOTION
MC_GroupInterrupt		COORDINATED MOTION
MC_GroupReadActualAcceleration		COORDINATED MOTION
MC_GroupReadActualPosition		COORDINATED MOTION
MC_GroupReadActualVelocity		COORDINATED MOTION
MC_GroupReadError		COORDINATED MOTION
MC_GroupReadStatus		COORDINATED MOTION
MC_GroupReset		COORDINATED MOTION
MC_GroupSetOverride		COORDINATED MOTION
MC_GroupSetPosition		COORDINATED MOTION
MC_GroupStop		COORDINATED MOTION
MC_Halt		MOTION CONTROL
MC_HaltSuperimposed		MOTION CONTROL
MC_Home		MOTION CONTROL
MC_MoveAbsolute		MOTION CONTROL
MC_MoveAdditive		MOTION CONTROL
MC_MoveCircularAbsolute		COORDINATED MOTION
MC_MoveCircularRelative		COORDINATED MOTION
MC_MoveContinuousAbsolute		MOTION CONTROL

Seznam pokračuje na další stránce...

Jméno bloku	Potřebná licence	
	STANDARD	Jiná
MC_MoveContinuousRelative		MOTION CONTROL
MC_MoveDirectAbsolute		COORDINATED MOTION
MC_MoveDirectRelative		COORDINATED MOTION
MC_MoveLinearAbsolute		COORDINATED MOTION
MC_MoveLinearRelative		COORDINATED MOTION
MC_MovePath		COORDINATED MOTION
MC_MovePath_PH		COORDINATED MOTION
MC_MoveRelative		MOTION CONTROL
MC_MoveSuperimposed		MOTION CONTROL
MC_MoveVelocity		MOTION CONTROL
MC_PhasingAbsolute		MOTION CONTROL
MC_PhasingRelative		MOTION CONTROL
MC_PositionProfile		MOTION CONTROL
MC_Power		MOTION CONTROL
MC_ReadActualPosition		MOTION CONTROL
MC_ReadAxisError		MOTION CONTROL
MC_ReadBoolParameter		MOTION CONTROL
MC_ReadCartesianTransform		COORDINATED MOTION
MC_ReadParameter		MOTION CONTROL
MC_ReadStatus		MOTION CONTROL
MC_Reset		MOTION CONTROL
MC_SetCartesianTransform		COORDINATED MOTION
MC_SetOverride		MOTION CONTROL
MC_Stop		MOTION CONTROL
MC_TorqueControl		MOTION CONTROL
MC_UngroupAllAxes		COORDINATED MOTION
MC_VelocityProfile		MOTION CONTROL
MC_WriteBoolParameter		MOTION CONTROL
MC_WriteParameter		MOTION CONTROL
MDL	•	
MDLI	•	
MID	•	
MINMAX	•	
ML_DGEBAK		MATRIX
ML_DGEBAL		MATRIX
ML_DGEBRD		MATRIX
ML_DGECON		MATRIX
ML_DGEES		MATRIX
ML_DGEEV		MATRIX

Seznam pokračuje na další stránce...

Jméno bloku	Potřebná licence	
	STANDARD	Jiná
ML_DGEHRD		MATRIX
ML_DGELQF		MATRIX
ML_DGELSD		MATRIX
ML_DGEQRF		MATRIX
ML_DGESDD		MATRIX
ML_DLACPY	•	
ML_DLANGE	•	
ML_DLASET	•	
ML_DTRSYL		MATRIX
MODULE	•	
MOSS		ADVANCED
MP	•	
MUL	•	
MVD	•	
MX_AT	•	
MX_ATSET	•	
MX_CNADD	•	
MX_CNMUL	•	
MX_CTODPA	•	
MX_DIM	•	
MX_DIMSET	•	
MX_DSAGET	•	
MX_DSAREF	•	
MX_DSASET	•	
MX_DTRNSP	•	
MX_DTRNSQ	•	
MX_FILL	•	
MX_FNX	•	
MX_MAT	•	
MX_RAND	•	
MX_REFCOPY	•	
MX_SLFS	•	
MX_VEC	•	
MX_WRITE	•	
MqttPublish		MQTT
MqttSubscribe		MQTT
NANINF	•	
NOT	•	
NSCL	•	

Seznam pokračuje na další stránce...

Jméno bloku	Potřebná licence	
	STANDARD	Jiná
NSSM		MODEL
NUREACT		MODEL
OPCUA	•	
OR	•	
ORHEXD	•	
OROCT	•	
ORQUAD	•	
OSCALL	•	
OSD	•	
OUTCONN	•	
OUTHEXD	•	
OUTOCT	•	
OUTQUAD	•	
OUTRHEXD		ADVANCED
OUTROCT		ADVANCED
OUTRQUAD		ADVANCED
OUTRSTD		ADVANCED
OUTSTD	•	
OpcUaReadValue		ADVANCED
OpcUaServerValue		ADVANCED
OpcUaWriteValue		ADVANCED
Outport	•	
PARA	•	
PARB	•	
PARE	•	
PARI	•	
PARR	•	
PARS	•	
PARX	•	
PCI		ADVANCED
PIDAT		AUTOTUNING
PIDE		ADVANCED
PIDGS		ADVANCED
PIDMA		AUTOTUNING
PIDMAW		AUTOTUNING
PIDMAX		AUTOTUNING
PIDU	•	
PIDUI		ADVANCED
PJROCT	•	

Seznam pokračuje na další stránce...

Jméno bloku	Potřebná licence	
	STANDARD	Jiná
PJSEXOCT	•	
PJSOCT	•	
POL	•	
POUT	•	
PRBS	•	
PRGM	•	
PROJECT	•	
PSD		ADVANCED
PSMPC		ADVANCED
PWM	•	
PYTHON		REXLANG
QCEDPOPT		ADVANCED
QCOPT		MODEL
QFC	•	
QFD	•	
QP_MPC2QP		ADVANCED
QP_OASES		ADVANCED
QP_UPDATE		ADVANCED
QTASK	•	
RAWCOPY		ADVANCED
RDC		ADVANCED
RDFT		ADVANCED
REC	•	
REGENP		ADVANCED
REL	•	
REPLACE	•	
REXLANG		REXLANG
RLIM	•	
RLY	•	
RM_AxesGroup		COORDINATED MOTION
RM_Axis		MOTION CONTROL
RM_AxisIn		MOTION CONTROL
RM_AxisOut		MOTION CONTROL
RM_AxisSpline		MOTION CONTROL
RM_DirectTorque		MOTION CONTROL
RM_DirectVelocity		MOTION CONTROL
RM_DriveMode		MOTION CONTROL
RM_Feed		COORDINATED MOTION
RM_Gcode		COORDINATED MOTION

Seznam pokračuje na další stránce...

Jméno bloku	Potřebná licence	
	STANDARD	Jiná
RM_GroupTrack		COORDINATED MOTION
RM_HomeOffset		MOTION CONTROL
RM_Track		MOTION CONTROL
RS	•	
RTOI	•	
RTOS	•	
RTOV	•	
S10F2		ADVANCED
SAI		ADVANCED
SAT	•	
SC2FA		AUTOTUNING
SCU	•	
SCUV	•	
SEL	•	
SELHEXD	•	
SELOCT	•	
SELQUAD	•	
SELSOCT	•	
SELU	•	
SETPA	•	
SETPB	•	
SETPI	•	
SETPR	•	
SETPS	•	
SETPX	•	
SG	•	
SGEN		MODEL
SGENTX		MODEL
SGI	•	
SGSLP		ADVANCED
SHIFTOCT	•	
SHLD	•	
SILO	•	
SILOS	•	
SINT	•	
SLEEP	•	
SMHCC		ADVANCED
SMHCCA		AUTOTUNING
SMTF		ADVANCED

Seznam pokračuje na další stránce...

Jméno bloku	Potřebná licence	
	STANDARD	Jiná
SOLNP		MATRIX
SOPDT	•	
SPIKE		ADVANCED
SPLIT_DT	•	
SQR	•	
SQRT	•	
SR	•	
SRTF		ADVANCED
SSW	•	
STATELOAD		ADVANCED
STATESAVE		ADVANCED
STEAM	•	
STMGEN		MODEL
STOR	•	
STR2DT	•	
STURB		MODEL
SUB	•	
SWR	•	
SWU	•	
SWVMR	•	
SYSEVENT	•	
SYSLOG	•	
SubSystem	•	
T2STR	•	
TASK	•	
TC	•	
TIME	•	
TIMER	•	
TIODRV	•	
TRIM	•	
TRND	•	
TRNDV	•	
TS	•	
TS2NS	•	
TSE	•	
TZ2UTC	•	
ToFile	•	
ToWorkspace	•	
UART	•	

Seznam pokračuje na další stránce...

Jméno bloku	Potřebná licence	
	STANDARD	Jiná
UNIPi_CHANNEL		UNIPi
UNIPi_PRODUCT		UNIPi
UNIPi_S1AI		UNIPi
UNIPi_S1AOR		UNIPi
UNIPi_S1CNT		UNIPi
UNIPi_S1DI		UNIPi
UNIPi_S1DO		UNIPi
UNIPi_S1LED		UNIPi
UNIPi_S1PWM		UNIPi
UNIPi_S2AI		UNIPi
UNIPi_S2AO		UNIPi
UNIPi_S2CNT		UNIPi
UNIPi_S2DI		UNIPi
UNIPi_S2RO		UNIPi
UTC2TZ	•	
UTOI	•	
VDEL	•	
VIN	•	
VOUT	•	
VTOR	•	
WASM		REXLANG
WEEK	•	
WSCH	•	
WWW	•	
ZV4IS		ADVANCED

Příloha C

Chybové kódy systému REXYGEN

Kódy úspěšných operací

- 0 V pořádku
- 1 Nepravda
- 2 První hodnota je větší
- 3 Druhá hodnota je větší
- 4 Parametr byl změněn
- 5 V pořádku, na serveru neprovedena žádná transakce
- 6 Příliš velká hodnota
- 7 Příliš malá hodnota
- 8 Operace probíhá
- 9 Upozornění ovladače systému REXYGEN
- 10 V archivu nejsou další položky
- 11 Položka je pole
- 12 Ukončeno
- 13 Konec souboru
- 14 Parametr pravděpodobně nesprávný

Obecné chybové kódy

- 100 Nedostatek paměti
- 101 Předpoklad nesplněn (Assertion failure)
- 102 Překročení času (timeout)
- 103 Obecná chyba vstupní proměnné
- 104 Nesprávná verze konfigurace
- 105 Není implementováno
- 106 Nesprávný parametr
- 107 Chyba služeb COM/OLE
- 108 Chyba modulu systému REXYGEN - některý ovladač nebo blok není nainstalován nebo licencován
- 109 Chyba ovladače systému REXYGEN

- 110 Úlohu operačního systému se nepodařilo vytvořit
- 111 Chyba volání funkce operačního systému
- 112 Nesprávná verze operačního systému
- 113 Přístup odmítnut operačním systémem
- 114 Perioda bloku nebyla nastavena
- 115 Selhala inicializace
- 116 Probíhá výměna konfigurace systému REXYGEN
- 117 Nesprávné cílové zařízení konfigurace
- 118 Přístup odmítnut systémem REXYGEN
- 119 Blok nebo jiný objekt není nainstalován nebo licencován
- 120 Kontrolní součty se liší
- 121 Objekt již existuje
- 122 Objekt neexistuje
- 123 Systémový uživatel nemá přiřazenou žádnou skupinu řídicího systému REXY-
GEN
- 124 Špatné heslo
- 125 Špatné uživatelské jméno nebo heslo
- 126 Cílové zařízení není kompatibilní
- 127 Zdroj nelze použít, neboť je uzamčen jiným modulem
- 128 Text není platný v kódování UTF8
- 129 Spuštění exekutivy není povoleno
- 130 Dosaženo k překročení maximálního počtu nějakého objektu
- 131 Došlo ke oříznutí textu
- 132 Nedostatečný buffer pro požadovanou operaci
- 133 Vykonávání bloku pozastaveno kvůli běhové chybě

Registrace tříd, chybové kódy symbolů a validačních procedur

- 200 Neregistrovaná třída
- 201 Třída už byla registrována
- 202 Nedostatek místa v registru
- 203 Index registru mimo rozsah
- 204 Nesprávný kontext
- 205 Nesprávný identifikátor
- 206 Nesprávný příznak vstupu
- 207 Nesprávná maska vstupu
- 208 Nesprávný druh objektu
- 209 Nesprávný typ proměnné
- 210 Nesprávný pracovní prostor objektu
- 211 Symbol nebyl nalezen
- 212 Symbol je nejednoznačný
- 213 Chyba kontroly rozsahu
- 214 Nedostatek místa pro hledání
- 215 Zápis do proměnné určené pouze pro čtení není dovolen
- 216 Data nejsou připravena

- 217 Hodnota mimo přípustný rozsah
- 218 Chyba připojení vstupu
- 219 Nalezena smyčka typů UNKNOWN
- 220 Chyba při překladu jazyka REXLANG

Kódy pro streamy a souborový systém

- 300 Přetečení streamu
- 301 Podtečení streamu
- 302 Vysílací chyba streamu
- 303 Příjímací chyba streamu
- 304 Chyba při posílání dat na cílové zařízení (download)
- 305 Chyba při posílání dat z cílového zařízení (upload)
- 306 Chyba vytvoření souboru
- 307 Chyba otvírání souboru
- 308 Chyba zavření souboru
- 309 Chyba čtení souboru
- 310 Chyba zápisu do souboru
- 311 Nesprávný formát
- 312 Chyba při komprimaci souborů
- 313 Chyba během extrahování souborů

Chyby komunikace

- 400 Chyba síťové komunikace
- 401 Komunikace už byla inicializována
- 402 Komunikace úspěšně ukončena
- 403 Nečekané zavření komunikace
- 404 Neznámý příkaz
- 405 Neočekávaný příkaz
- 406 Nečekané zavření komunikace, pravděpodobně 'příliš mnoho klientů'
- 407 Překročení časového limitu pro komunikaci (timeout)
- 408 Cílové zařízení nebylo nalezeno
- 409 Spojení selhalo
- 410 Konfigurace systému REXYGEN byla změněna
- 411 Běh exekutivy systému REXYGEN se ukončuje
- 412 Běh exekutivy systému REXYGEN byl ukončen
- 413 Spojení odmítnuto
- 414 Cílové zařízení není dostupné
- 415 Cílové zařízení nebylo nalezeno v záznamu DNS
- 416 Chyba při čtení ze soketu
- 417 Chyba zápisu do soketu
- 418 Chybná operace na soketu
- 419 Rezervováno pro soket 1
- 420 Rezervováno pro soket 2
- 421 Rezervováno pro soket 3

- 422 Rezervováno pro soket 4
- 423 Rezervováno pro soket 5
- 424 Nelze vytvořit kontext SSL
- 425 Nelze načíst certifikát
- 426 Chyba při vyjednávání spojení SSL
- 427 Chyba verifikace certifikátu
- 428 Rezervováno pro SSL 2
- 429 Rezervováno pro SSL 3
- 430 Rezervováno pro SSL 4
- 431 Rezervováno pro SSL 5
- 432 Relace odmítnuta
- 433 STARTTLS odmítnuto
- 434 Ověřovací metoda odmítnuta
- 435 Ověření selhalo
- 436 Chyba operace vysílání
- 437 Chyba operace přijímání
- 438 Komunikační příkaz selhal
- 439 Vyrovnávací paměť pro příjem je příliš malá
- 440 Vyrovnávací paměť pro vysílání je příliš malá
- 441 Špatná hlavička
- 442 Server HTTP vrátil chybu
- 443 Server HTTP vrátil přesměrování
- 444 Nepřípustná blokující operace
- 445 Neplatná operace
- 446 Komunikace ukončena
- 447 Připojování přerušeno

Kódy numerických chyb

- 500 Obecná numerická chyba
- 501 Dělení nulou
- 502 Přetečení numerického zásobníku
- 503 Neplatná numerická instrukce
- 504 Neplatná numerická adresa
- 505 Nesprávný numerický typ
- 506 Neinicializovaná numerická hodnota
- 507 Přetečení/podtečení numerického argumentu
- 508 Numerická chyba kontroly rozsahu
- 509 Nesprávný rozsah indexů vektoru/matice
- 510 Číselná hodnota příliš blízká nule

Kódy archivního systému

- 600 Chyba prohledávání archivu
- 601 Fatální chyba archivního semaforu
- 602 Archiv byl smazán

- 603 Archiv byl rekonstruován ze záložních proměnných
- 604 Archiv byl rekonstruován z normálních proměnných
- 605 Chyba kontrolního součtu archivu
- 606 Chyba integrity archivu
- 607 Byla změněna velikost archivu
- 608 Byla překročena povolená velikost archivu

Kódy bloků pro řízení pohybu

- 700 MC - Neplatný parametr
- 701 MC - Mimo rozsah
- 702 MC - Pozice není dosažitelná
- 703 MC - Neplatný stav osy
- 704 MC - Překročen limit momentu
- 705 MC - Překročen časový limit
- 706 MC - Překročena hraniční pozice
- 707 MC - Skoková změna pozice nebo rychlosti
- 708 MC - Base axis error or invalid state
- 709 MC - Pohyb zastaven vstupem FAULT
- 710 MC - Pohyb zastaven polohou mimo rozsah osy
- 711 MC - Pohyb zastaven z důvodu překročení maximální rychlosti osy
- 712 MC - Pohyb zastaven z důvodu překročení maximálního zrzchlení osy
- 713 MC - Pohyb zastaven koncovým spínačem
- 714 MC - Pohyb zastaven z důvodu překročení maximální odchylky polohy (LAG)
- 715 MC - Osa deaktivována během pohybu
- 716 MC - Chyba generování přechodové křivky
- 717 MC - Chyba servoměniče
- 718 MC - nepoužito
- 719 MC - nepoužito
- 720 MC - Obecná chyba
- 721 MC - Není implementováno
- 722 MC - Příkaz ukončen
- 723 MC - Rozdílná perioda osy a bloku
- 724 MC - Blok čeká na převzetí osy

Kódy licencovacího systému

- 800 Nepodařila se identifikace síťového rozhraní
- 801 Nepodařila se identifikace CPU
- 802 Nepodařila se identifikace HDD
- 803 Neplatný kód zařízení
- 804 Neplatný licenční klíč
- 805 Licence nenalezena

Kódy spojené s webserverem

- 900 Příliš rozsáhlý požadavek na webový server
- 901 Příliš rozsáhlá odpověď webového serveru
- 902 Neplatný formát
- 903 Neplatný parametr

Kódy spojené s knihovnou RexVision

- 1000 ... Výsledek není vyhodnocen
- 1001 ... Nelze nalézt hledaný objekt/vzor
- 1002 ... Zadanému kritériu hledání vyhovuje více objektů

Kódy spojené se standardem FMI

- 1100 ... Nepodařilo se alokovat kontext FMI
- 1101 ... Nesprávná verze FMU
- 1102 ... Chyba parsování souboru XML pro FMI
- 1103 ... Vyžadován pouze druh FMI pro Model Exchange
- 1104 ... Vyžadován pouze druh FMI pro Co-Simulation
- 1105 ... Nepodařilo se zavést FMU
- 1106 ... Nepodařilo se vytvořit instanci FMU
- 1107 ... Nepodařilo se ukončit instanci FMU
- 1108 ... Selhal reset FMU
- 1109 ... Selhalo nastavení experimentu FMU
- 1110 ... Selhalo zahájení inicializačního módu FMU
- 1111 ... Selhalo ukončení inicializačního módu FMU
- 1112 ... Chyba získání seznamu proměnných FMU
- 1113 ... Chyba čtení reálné proměnné z FMU
- 1114 ... Chyba zápisu reálné proměnné do FMU
- 1115 ... Chyba čtení celočíselné proměnné z FMU
- 1116 ... Chyba zápisu celočíselné proměnné do FMU
- 1117 ... Chyba čtení booleovské proměnné z FMU
- 1118 ... Chyba zápisu booleovské proměnné do FMU
- 1119 ... Chyba provedení simulačního kroku FMU
- 1120 ... FMU má příliš mnoho vstupů
- 1121 ... FMU má příliš mnoho výstupů
- 1122 ... FMU má příliš mnoho parametrů

Příloha D

Speciální signály systému REXYGEN

Dále je uveden seznam speciálních signálů, které lze číst uvnitř REXYGENu. Podrobnosti o jejich použití najdete v příkladu *0001_Special_Signals*.

Parameter	Popis	Data Type	Možnosti
perf	Frekvence výkonnostního čítače	LARGE	EXEC
period	Perioda	DOUBLE	LEVEL, TASK, SEQ, IODRV, QTASK, BLOCK, ARCHIVE
nblocks	Počet bloků	SHORT	TASK, SEQ, QTASK
stack	Velikost zásobníku	LONG	TASK, IODRV, QTASK
exfac	Faktor provedení tasku	DWORD	TASK, IODRV, QTASK
start	Start tick	DWORD	TASK
stop	Stop tick	DWORD	TASK
ntasks	Počet tasků	SHORT	LEVEL, IODRV, EXEC
ntick	Počet ticků levelu	DWORD	LEVEL
pri	Priorita levelu	SHORT	LEVEL, IODRV, QTASK
tick	Hlavní tick	LARGE	EXEC
nlevels	Počet levelů	SHORT	EXEC
nmodules	Počet modulů	SHORT	EXEC
ndrivers	Počet ovladačů	SHORT	EXEC
narchives	Počet archivů	SHORT	EXEC
nqtasks	Počet rychlých tasků	SHORT	EXEC
tcomp	Čas kompilace [ns od epochy]	LARGE	EXEC
tdnld	Čas downloadu [ns od epochy]	LARGE	EXEC
bufsize	Velikost bufferu (archiv)	LONG	ARCHIVE
timesize	Velikost indexového bufferu (archiv)	LONG	ARCHIVE
daysize	Velikost denního bufferu (archiv)	LARGE	ARCHIVE
errblk	Index bloku s chybou	SHORT	TASK, SEQ, QTASK
errno	Kód chyby	SHORT	TASK, SEQ, QTASK, IODRV
status	Kód stavu (ovladač)	LONG	IODRV
over	Počet překryvů/kolizí (qtask)	LARGE	QTASK, TASK, IODRV
excnt	Počet spuštění tasku	LARGE	LEVEL, TASK, SEQ, IODRV, QTASK
tlast	Číslo posledního provedení [tický výkonnostního čítače]	LARGE	LEVEL, TASK, SEQ, IODRV, QTASK
tmin	Minimální čas provedení [tický výkonnostního čítače]	LARGE	LEVEL, TASK, SEQ, IODRV, QTASK
tmax	Maximální čas provedení [tický výkonnostního čítače]	LARGE	LEVEL, TASK, SEQ, IODRV, QTASK
tsum	Součet provedení [ns nebo ticky]	LARGE	LEVEL, TASK, SEQ, IODRV, QTASK
tavg	Průměrný čas provedení (tsum/excnt)[ticky výkonnostního čítače]	LARGE	LEVEL, TASK, SEQ, IODRV, QTASK
dstart	Časový údaj o zahájení [tický výkonnostního čítače]	LARGE	IODRV, TASK
dstop	Časový údaj o ukončení [tický výkonnostního čítače]	LARGE	IODRV, TASK
tover	Overtime [tický výkonnostního čítače]	LARGE	TASK
dmstart	Maximální start delay [tický výkonnostního čítače]	LARGE	IODRV, TASK
dmstop	Maximální stop delay [tický	LARGE	IODRV, TASK

Literatura

- [1] Unipi Technology s.r.o. Unipi technology. <http://www.unipi.technology>, 2024. →.
- [2] OPC Foundation. *Data Access Custom Interface Specification Version 3.00*. OPC Foundation, P.O. Box 140524, Austin, Texas, USA, 2003.
- [3] REX Controls s.r.o.. *OPC UA server pro systém REXYGEN – Referenční manuál*, 2019. →.
- [4] REX Controls s.r.o.. *REXYGEN Studio – User manual*, 2024. →.
- [5] Schlegel Miloš. Fuzzy regulátor: tutoriál. *www.rexcontrols.cz*, pages 1–10, 2002.
- [6] M. Goubelj. Kalman filter based observer design for real-time frequency identification in motion control systems. <https://ieeexplore.ieee.org/document/7169979>, 2015.
- [7] Miloš Schlegel, Pavel Balda, and Milan Štětina. Robustní PID autotuner: momentová metoda. *Automatizace*, 46(4):242–246, 2003.
- [8] M. Schlegel and P. Balda. Diskretizace spojitého lineárního systému (in Czech). *Automatizace*, 11, 1987.
- [9] J. Königsmarková and M. Schlegel. Identification of n-link inverted pendulum on a cart. <https://ieeexplore.ieee.org/document/7976186>, 2017.
- [10] BLAS 3.8.0. – netlib.org. Basic Linear Algebra Subprograms Version 3.8.0. <http://www.netlib.org/blas/>, 2017.
- [11] LAPACK 3.8.0 – netlib.org. Linear Algebra PACKage Version 3.8.0. <http://www.netlib.org/lapack/explore-html/index.html>, 2018.
- [12] Cleve Moler and Charles Van Loan. Nineteen dubious ways to compute the exponential of a matrix, twenty-five years later. *SIAM Review*, 45(1):3–49, 2003.
- [13] J.M. Maciejowski. *Predictive Control with Constraints*. Prentice Hall, England., 2002.
- [14] H.J. Ferreau, C. Kirches, A. Potschka, H.G. Bock, and M. Diehl. qpOASES: A parametric active-set algorithm for quadratic programming. *Mathematical Programming Computation*, 6(4):327–363, 2014.

- [15] H.J. Ferreau. *qpOASES User's Manual, Version 3.2 (April 2017)*, 2017.
- [16] Y. Ye. *Interior algorithms for linear quadratic and linearly constrained nonlinear programming*. Ph.d. dissertation, Dept. Elect. Eng. Syst., Stanford Univ., Stanford, CA, 1987.
- [17] Y. Ye. *User's Manual for Matlab Linear Programming Codes*, 1989. →.
- [18] LabLua PUC-Rio. Lua 5.4 Reference Manual. <https://www.lua.org/manual/5.4/>, 2020.
- [19] Python Software Foundation. Python documentation. <https://docs.python.org/3/>, 2019.
- [20] Martin Reinecke. PocketFFT. <https://gitlab.mpcdf.mpg.de/mtr/pocketfft>, 2019.
- [21] Paul N. Swarztrauber. FFTPACK Version 4 – netlib.org. <https://www.netlib.org/fftpack>, 1985.
- [22] Paul N. Swarztrauber. Vectorizing the ffts**this chapter was written while the author was visiting the scientific computing division at the national bureau of standards. In GARRY RODRIGUE, editor, *Parallel Computations*, pages 51–83. Academic Press, 1982.
- [23] P. Welch. The use of fast fourier transform for the estimation of power spectra: A method based on time averaging over short, modified periodograms. *IEEE Transactions on Audio and Electroacoustics*, 15(2):70–73, 1967.
- [24] Wikipedia. Welch's method. https://en.wikipedia.org/wiki/Welch's_method.
- [25] REX Controls s.r.o.. *Ovladač MQTTDrv systému REXYGEN – Uživatelská příručka*, 2020. →.
- [26] OASIS. MQTT Version 3.1.1. <http://docs.oasis-open.org/mqtt/mqtt/v3.1.1/os/mqtt-v3.1.1-os.html>, 2014.
- [27] REX Controls s.r.o.. *Ovladač OpcUaDrv systému REXYGEN – Uživatelská příručka*, 2024. →.
- [28] REX Controls s.r.o.. *Licencování REXYGEN – Uživatelská příručka*, 2024. →.

Rejstřík

úloha

doba provádění, 59

priorita, 59

rychlá, 48

čítání pulsů

obousměrné, 309

časovač

systémový, 42

šířka pásma, 164

ABS, 18, 26, 100, 101, 963

ABSR0T, 26, 143, 963

ACD, 341, 344, 963

ACLEAR, 341, 346, 963

ADD, 18, 26, 100, 102, 103, 963

ADDEXD, 26, 103, 963

ADDOCT, 102, 103, 138, 963

ADDQUAD, 103, 963

AFLUSH, 341, 347, 963

ALARMS, 29, 348, 350, 351, 963

ALB, 29–31, 341, 348, 354, 963

ALBI, 29–31, 341, 348, 354, 963

ALM, 29–31, 341, 350, 963

ALMI, 29–31, 341, 350, 963

ALN, 29–31, 341, 351, 354, 963

ALNI, 29–31, 341, 351, 354, 963

AND, 18, 26, 301–303, 963

ADDEXD, 26, 302, 303, 963

ANDDOCT, 302, 303, 963

ANDQUAD, 302, 303, 963

ANLS, 18, 195, 196, 964

aplikace

řídícího systému REXYGEN, 35

ARC, 32, 342, 347, 964

architektura

otevřená, 44

archiv, 341

alarmů, 32

konfigurace, 32

na disku, 342

RAM, 341

trendů, 32

událostí, 32

v zálohované paměti, 342

ARLY, 26, 211, 964

ARS, 341, 354, 964

ASW, 18, 26, 142, 145, 964

ATMT, 18, 26, 301, 304, 311, 386, 401, 408, 964

AVG, 18, 26, 142, 147, 964

AVS, 26, 148, 964

AVSI, 26, 149, 964

běh úloh, 50

BDHEXD, 26, 306, 311, 964

BDOCT, 18, 301, 306, 311, 964

Besselův filtr, 164

BINS, 18, 195, 198, 200, 202, 964

BIS, 18, 195, 198, 200, 204, 964

BISR, 18, 195, 202, 964

BITOP, 18, 26, 139, 301, 307, 964

blok

formát popisu, 22

parametry, 23

popis funkce, 23

symbol, 22

výstupu, 23

vstupy, 23

BMHEXD, 26, 308, 311, 964

BMOCT, 18, 301, 308, 311, 964

BPF, 18, 26, 142, 152, 964

BSFIFO, 20, 631, 632, 964

BSGET, 20, 631, 634, 636, 637, 964
 BSGETOCT, 634, 637, 964
 BSGETOCTV, 636, 638, 964
 BSGETV, 20, 631, 636, 638, 964
 BSSET, 20, 631, 634, 637, 964
 BSSETOCT, 634, 637, 964
 BSSETOCTV, 636, 638, 964
 BSSETV, 20, 631, 636, 638, 964
 Butterworthův filtr, 164

CanItem, 22, 907, 908, 965
 CanRecv, 22, 907–909, 965
 CanSend, 22, 907, 911, 965
 CDELSSM, 19, 26, 416, 417, 964

cesta

úplná, 50

chyba

fatální, 48

CMP, 26, 153, 964
 CNA, 26, 385, 400, 460, 964
 CNB, 26, 104, 964
 CNDR, 26, 154, 964
 CNDT, 964
 CNE, 105, 964
 CNI, 26, 106, 964
 CNR, 26, 107, 964
 CNS, 19, 26, 361, 362, 964
 CNT, 964
 CONCAT, 19, 26, 361, 363, 964
 CONCAT_DT, 965
 COUNT, 18, 26, 301, 309, 965
 CSSM, 19, 26, 416, 420, 965

DATA, 34, 965
 DATE, 19, 327–329, 965
 DATETIME, 19, 327–329, 334, 965
 DDELSSM, 19, 26, 416, 423, 965
 DEL, 18, 25, 26, 142, 156, 965
 DELM, 26, 157, 965
 DER, 26, 158, 965
 DFIR, 19, 26, 416, 425, 965
 DIF, 26, 108, 965
 Display, 67, 965
 DIV, 18, 25, 26, 100, 109, 129, 965

doba

provádění úlohy, 59

DSSM, 19, 26, 416, 426, 965

DT2STR, 965

EAS, 26, 110, 965

EATMT, 26, 311, 965

EDGE, 26, 181, 314, 965

EKF, 19, 416, 428, 441, 965

EMD, 26, 111, 965

EPC, 20, 46, 579, 580, 965

EQ, 26, 315, 965

EVAR, 26, 159, 965

EXEC, 17, 28, 32, 34, 35, 37, 40, 42, 44, 48,
 49, 59, 168, 332, 965

FFT, 20, 631, 639, 965

filtr

šířka pásma, 164

Besselův, 164

Butterworthův, 164

nelineární, 186

filtrace

číslicová vstupních signálů, 48

FIND, 19, 26, 361, 364, 965

FIWR, 26, 212, 965

FLCU, 18, 26, 210, 217, 965

FMUCS, 19, 416

FMUINFO, 19, 416

FNX, 18, 26, 100, 112, 965

FNXY, 18, 26, 100, 114, 965

FOPDT, 450

FOPDT, 19, 26, 241, 248, 416, 431, 965

frekvenční charakteristika, 232

FRID, 26, 220, 965

From, 71, 75, 85, 965

FromFile, 69, 965

FromWorkspace, 70, 965

GAIN, 26, 116, 966

generátor

časových funkcí, 260

GETPA, 19, 26, 72, 87, 383, 384, 387, 400,
 966

GETPB, 386, 388, 389, 966

- GETPI, 386, 388, 389, 598, 599, 966
- GETPR, 26, 72, 386, 388, 389, 408, 966
- GETPS, 19, 26, 72, 383, 388, 389, 966
- GETPX, 26, 387–389, 966
- Goto, 71, 75, 85, 909, 911, 966
- GotoTagVisibility, 71, 86, 966
- GRADS, 117, 966
- HID, 966
- HMI, 17, 28, 37, 966
- hodnota
 - implicitní, 23
 - maximální, 23
 - minimální, 23
 - náhradní, 109, 111, 112, 114, 121, 122, 132, 137
- HTTP, 583, 966
- HTTP2, 20, 579, 583, 585, 966
- I3PM, 224, 966
- IADD, 18, 26, 100, 119, 966
- IDIV, 18, 26, 100, 121, 966
- IM201CNT, 923, 966
- IM201DI, 924, 966
- IM203DO, 925, 966
- IM203PWM, 926, 966
- IM204CNT, 927, 966
- IM204DI, 928, 966
- IM205DO, 929, 966
- IM205PWM, 930, 966
- IM301CNT, 932, 966
- IM301DI, 933, 966
- IM301DO, 934, 966
- IM502AO, 935, 966
- IM503AI, 936, 966
- IM504RI, 938, 966
- IM506AI, 940, 966
- IM506AO, 942, 966
- IMOD, 18, 26, 100, 122, 966
- IMUL, 18, 26, 100, 123, 966
- INCONN, 72, 385, 387, 388, 396, 966
- INFO, 39, 966
- INHEXD, 73, 966
- inicializace
 - pořadí modulů, 40
 - pořadí ovladačů, 40
 - rychlé úlohy, 48
- INOCT, 73, 77, 966
- Inport, 87, 91, 387, 402, 755, 967
- INQUAD, 73, 967
- INSTD, 73, 75, 77, 79, 967
- INTE, 26, 160, 185, 967
- INTSM, 26, 139, 316, 967
- IOASYNC, 26, 77, 967
- IODRV, 40, 75, 85, 967
- IOTASK, 42, 50, 62, 384, 387, 399, 402, 418, 420, 967
- IPEN2, 432, 435, 967
- IPEN2p, 967
- IPEN2pu, 432, 435, 967
- IPEN3, 432, 435, 967
- IPEN3p, 967
- IPEN3pu, 432, 435, 967
- IRIS_MODULE, 943, 967
- ISSW, 26, 317, 967
- ISUB, 26, 125, 967
- ITOI, 26, 318, 967
- ITOS, 19, 26, 361, 365, 967
- jmenovatel, 111
- KDER, 26, 162, 641, 967
- kompatibilia
 - REXYGEN a Simulink, 43
- konfigurace
 - archivy, 35
 - moduly, 35
 - systému REXYGEN, 35
 - výpočetní úloha, 35
 - vstupně-výstupní ovladače, 35
- LC, 26, 226, 967
- LEN, 19, 26, 361, 366, 967
- LIN, 18, 26, 100, 127, 967
- LLC, 26, 227, 450, 967
- LPBRK, 26, 43, 145, 967
- LPF, 26, 164, 967
- LPI, 26, 228, 967
- LUA, 620, 967

- LUAHEXD, 620, 967
 LUAOCT, 620, 967
 LUAQUAD, 620, 967

 MB_DASUM, 461, 967
 MB_DAXPY, 463, 967
 MB_DCOPY, 465, 967
 MB_DDOT, 467, 967
 MB_DGEMM, 20, 459, 469, 967
 MB_DGEMV, 471, 967
 MB_DGER, 20, 459, 473, 967
 MB_DNRM2, 475, 515, 967
 MB_DROT, 477, 968
 MB_DSCAL, 479, 968
 MB_DSWAP, 481, 968
 MB_DTRMM, 20, 459, 483, 968
 MB_DTRMV, 485, 968
 MB_DTRSV, 487, 968
 MC_AccelerationProfile, 21, 658, 687, 969
 MC_AddAxisToGroup, 21, 803, 831, 894, 897, 969
 MC_CamIn, 21, 762, 766, 777, 781, 796, 799, 969
 MC_CamOut, 21, 762, 777, 781, 969
 MC_CombineAxes, 21, 761, 783, 969
 MC_CombineAxis, 794, 796, 799
 MC_GearIn, 21, 761, 786, 789, 794, 796, 799, 969
 MC_GearInPos, 21, 762, 789, 794, 796, 799, 969
 MC_GearOut, 21, 761, 786, 794, 969
 MC_GroupContinue, 832, 841, 969
 MC_GroupDisable, 21, 803, 833, 894, 969
 MC_GroupEnable, 21, 803, 834, 894, 897, 969
 MC_GroupHalt, 21, 803, 835, 969
 MC_GroupInterrupt, 832, 841, 969
 MC_GroupReadActualAcceleration, 842, 897, 969
 MC_GroupReadActualPosition, 21, 803, 843, 897, 969
 MC_GroupReadActualVelocity, 22, 803, 844, 897, 969
 MC_GroupReadError, 22, 803, 845, 969
 MC_GroupReadStatus, 22, 803, 846, 969
 MC_GroupReset, 848, 894, 969
 MC_GroupSetOverride, 849, 969
 MC_GroupSetPosition, 852, 969
 MC_GroupStop, 836, 854, 969
 MC_Halt, 21, 659, 691, 969
 MC_HaltSuperimposed, 693, 969
 MC_Home, 21, 658, 694, 969
 MC_MoveAbsolute, 21, 658, 696, 703, 732, 759, 790, 969
 MC_MoveAdditive, 699, 969
 MC_MoveCircularAbsolute, 21, 803, 857, 969
 MC_MoveCircularRelative, 21, 803, 862, 969
 MC_MoveContinuousAbsolute, 702, 969
 MC_MoveContinuousRelative, 706, 970
 MC_MoveDirectAbsolute, 21, 803, 867, 904, 970
 MC_MoveDirectRelative, 21, 803, 871, 970
 MC_MoveLinearAbsolute, 21, 803, 836, 839, 849, 855, 875, 894, 899, 970
 MC_MoveLinearRelative, 21, 803, 879, 970
 MC_MovePath, 21, 803, 883, 970
 MC_MovePath_PH, 886, 970
 MC_MoveRelative, 21, 658, 707, 710, 713, 970
 MC_MoveSuperimposed, 693, 713, 796, 799, 970
 MC_MoveVelocity, 21, 658, 716, 970
 MC_PhasingAbsolute, 21, 762, 796, 970
 MC_PhasingRelative, 21, 762, 799, 970
 MC_PositionProfile, 21, 659, 719, 759, 765, 970
 MC_Power, 21, 658, 723, 970
 MC_ReadActualPosition, 21, 659, 724, 970
 MC_ReadAxisError, 21, 659, 725, 970
 MC_ReadBoolParameter, 726, 970
 MC_ReadCartesianTransform, 22, 803, 888, 970
 MC_ReadParameter, 21, 659, 727, 970
 MC_ReadStatus, 729, 970
 MC_Reset, 21, 659, 731, 848, 970
 MC_SetCartesianTransform, 22, 803, 888, 890, 970

- MC_SetCartesianTransforms, 897
- MC_SetKinTransform_Lin, 857, 862, 867, 871, 875, 879, 894
- MC_SetOverride, 732, 970
- MC_SetPosition, 694
- MC_Stop, 21, 659, 734, 970
- MC_TorqueControl, 736, 747, 970
- MC_UngroupAllAxes, 893, 970
- MC_UngroupAllAxis, 894, 897
- MC_VelocityProfile, 21, 659, 739, 970
- MC_WriteBoolParameter, 743, 970
- MC_WriteParameter, 21, 659, 744, 970
- MCP_AccelerationProfile, 660, 687, 968
- MCP_CamIn, 763, 765, 777, 968
- MCP_CamTableSelect, 21, 762, 765, 777, 778, 968
- MCP_CombineAxes, 767, 783, 968
- MCP_GearIn, 769, 786, 968
- MCP_GearInPos, 771, 789, 968
- MCP_GroupHalt, 805, 835, 968
- MCP_GroupInterrupt, 806, 841, 968
- MCP_GroupSetOverride, 807, 849, 968
- MCP_GroupSetPosition, 808, 852, 968
- MCP_GroupStop, 809, 854, 968
- MCP_Halt, 662, 691, 968
- MCP_HaltSuperimposed, 663, 693, 968
- MCP_Home, 664, 694, 968
- MCP_MoveAbsolute, 666, 696, 968
- MCP_MoveAdditive, 668, 699, 968
- MCP_MoveCircularAbsolute, 810, 857, 968
- MCP_MoveCircularRelative, 812, 862, 968
- MCP_MoveContinuousAbsolute, 670, 702, 968
- MCP_MoveContinuousRelative, 672, 706, 968
- MCP_MoveDirectAbsolute, 814, 867, 968
- MCP_MoveDirectRelative, 816, 871, 968
- MCP_MoveLinearAbsolute, 818, 875, 968
- MCP_MoveLinearRelative, 820, 879, 968
- MCP_MovePath, 822, 883, 968
- MCP_MovePath_PH, 824, 968
- MCP_MoveRelative, 674, 710, 968
- MCP_MoveSuperimposed, 676, 713, 968
- MCP_MoveVelocity, 677, 716, 968
- MCP_PhasingAbsolute, 773, 796, 968
- MCP_PhasingRelative, 775, 799, 968
- MCP_PositionProfile, 679, 719, 968
- MCP_SetCartesianTransform, 826, 890, 968
- MCP_SetKinTransform_Arm, 827, 969
- MCP_SetKinTransform_Lin, 969
- MCP_SetKinTransform_Schunk, 969
- MCP_SetKinTransform_UR, 829, 969
- MCP_SetOverride, 681, 732, 969
- MCP_Stop, 682, 734, 969
- MCP_TorqueControl, 683, 736, 969
- MCP_VelocityProfile, 685, 739, 969
- MCU, 26, 230, 298, 969
- MDL, 19, 26, 416, 437, 438, 970
- MDLI, 19, 26, 416, 438, 970
- metoda nejmenších čtverců, 158
- MID, 19, 26, 361, 367, 970
- MINMAX, 26, 166, 970
- ML_DGEBAK, 489, 970
- ML_DGEBAL, 492, 970
- ML_DGEBRD, 20, 459, 494, 970
- ML_DGECON, 496, 970
- ML_DGEES, 498, 970
- ML_DGEES , 20, 459
- ML_DGEEV, 20, 459, 500, 970
- ML_DGEHRD, 502, 971
- ML_DGELQF, 505, 971
- ML_DGELSD, 507, 971
- ML_DGEQRF, 20, 459, 509, 971
- ML_DGESDD, 20, 459, 511, 971
- ML_DLACPY, 513, 971
- ML_DLANGE, 475, 515, 971
- ML_DLASET, 517, 971
- ML_DTRSYL, 519, 971
- model
 - FOPDT, 450
- modul, 44
 - rozšiřující, 40
- MODULE, 40, 44, 971
- MOSS, 20, 631, 641, 971
- MP, 18, 195, 204, 348, 351, 971
- MqttPublish, 21, 651, 652, 971
- MqttSubscribe, 21, 651, 654, 971
- MUL, 18, 26, 100, 128, 971
- MVD, 19, 26, 416, 439, 971
- MX_AT, 521, 971

- MX_ATSET, 522, 971
- MX_CNADD, 523, 971
- MX_CNMUL, 524, 971
- MX_CTODPA, 525, 971
- MX_DIM, 527, 971
- MX_DIMSET, 528, 971
- MX_DSAGET, 530, 971
- MX_DSAREF, 532, 971
- MX_DSASET, 534, 971
- MX_DTRNSP, 20, 459, 536, 971
- MX_DTRNSQ, 538, 971
- MX_FILL, 20, 459, 540, 971
- MX_FNX, 541, 971
- MX_MAT, 20, 385, 400, 459, 543, 561, 571, 971
- MX_RAND, 20, 459, 544, 971
- MX_REFCOPY, 546, 971
- MX_SLFS, 547, 971
- MX_VEC, 20, 459, 550, 971
- MX_WRITE, 551, 971

- NANINF, 26, 129, 971
- NOT, 18, 26, 301, 319, 971
- NSCL, 26, 167, 971
- NSSM, 429, 440, 972
- NUREACT, 443, 972

- OPCUA, 45, 972
- OpcUaReadValue, 22, 913, 914, 972
- OpcUaServerValue, 22, 913, 916, 972
- OpcUaWriteValue, 22, 913, 918, 972
- operace
 - binární, 133
 - bitová, 307
- OR, 18, 26, 301, 320, 321, 972
- ORHEXD, 26, 320, 321, 972
- OROCT, 320, 321, 972
- ORQUAD, 320, 321, 972
- OSCALL, 46, 581, 972
- OSD, 26, 168, 972
- OUTCONN, 78, 400, 402, 404, 972
- OUTHEXD, 79, 82, 972
- OUTOCT, 77, 79, 82, 972
- Outport, 87, 91, 387, 402, 755, 972
- OUTQAD, 909, 911
- OUTQUAD, 79, 82, 972
- OUTRHEXD, 82, 972
- OUTROCT, 82, 972
- OUTRQUAD, 82, 972
- OUTRSTD, 84, 972
- OUTSTD, 73, 77, 79, 84, 85, 909, 911, 972
- ovladač
 - konfigurační data, 40
 - pořadí inicializace, 40
 - soubor s příponou .rio, 40
 - systém REXYGEN, 40
 - uživatelská dokumentace, 42
 - vstupně-výstupní, 40
- pásmo propustnosti, 152
- překlad
 - program REXYGEN Compiler, 43
- překladač REXYGEN Compiler, 35
- PARAM, 391, 972
- PARAM, 605
- parametr
 - tick, 35
 - vzdálené získávání, 386
- PARB, 394, 396, 397, 972
- PARE, 393, 972
- PARI, 393, 394, 396, 397, 972
- PARR, 19, 183, 383, 394, 396, 397, 972
- PARS, 396, 397, 972
- PARX, 394, 396, 397, 972
- PCI, 643, 972
- PIDAT, 18, 26, 210, 232, 972
- PIDE, 26, 235, 972
- PIDGS, 18, 26, 210, 237, 972
- PIDMA, 18, 26, 210, 224, 239, 246, 278, 588, 972
- PIDMAW, 972
- PIDMAX, 26, 246, 972
- PIDU, 26, 232, 235, 237, 239, 253, 257, 278, 281, 298, 588, 972
- PIDUI, 26, 257, 972
- PJROCT, 19, 361, 368, 972
- PJSEXOCT, 370, 973
- PJSOCT, 370, 371, 973

- pořadí
 - inicializace úloh, 59
 - inicializace modulů, 44
 - spouštění úloh, 59
 - zavádění modulů, 44
- podsystem
 - archivační, 19, 341
- POL, 18, 26, 100, 131, 973
- POUT, 26, 259, 973
- PRBS, 18, 195, 205, 973
- PRGM, 260, 973
- priorita
 - úloh, 59
 - logická, 35, 40, 48
 - závislost na operačním systému, 35
- program
 - REXYGEN Compiler, 43
 - REXYGEN Studio, 32, 40, 48, 59, 62, 87, 385, 387, 400, 402
 - REXYGEN Studio, příznak Enable, 50
 - REXYGEN Studio, tlačítko Halt/Run, 50
 - REXYGEN Studio, tlačítko RESET, 51
- PROJECT, 47, 973
- projekt
 - hlavní soubor, 35, 40
- protokol
 - UDP/IP, 587
- první řád, 450
- PSD, 20, 631, 644, 973
- PSMPC, 26, 224, 262, 973
- pulz, 259
- PWM, 26, 240, 247, 254, 266, 287, 291, 292, 973
- PYTHON, 616, 624, 634, 973
- QCEDPOPT, 20, 557, 973
- QCOPT, 444, 973
- QFC, 89, 90, 597, 973
- QFD, 82, 84, 89, 90, 597, 973
- QP_MPC2QP, 20, 557, 558, 973
- QP_OASES, 20, 557, 558, 565, 973
- QP_UPDATE, 20, 557, 558, 570, 973
- QTASK, 17, 28, 35, 42, 48, 50, 59, 418, 420, 583, 585, 611, 973
- Rate monotonic scheduling, 36
- RAWCOPY, 650, 973
- RDC, 20, 579, 587, 973
- RDFT, 26, 169, 973
- režie
 - jádra řídicího systému, 35
- REC, 26, 132, 973
- REGEXP, 19, 361, 373, 973
- REL, 26, 133, 973
- REPLACE, 19, 26, 361, 376, 973
- REXLANG, 20, 355, 429, 440, 441, 575, 579, 590, 616, 619, 620, 624, 634, 973
- RLIM, 26, 171, 973
- RLY, 26, 211, 268, 973
- RM_AxesGroup, 831, 833, 834, 845, 894, 973
- RM_Axis, 555, 694, 724–727, 732, 743–745, 752, 754, 755, 831, 833, 849, 894, 897, 973
- RM_AxisGroup, 893
- RM_AxisIn, 973
- RM_AxisOut, 752, 973
- RM_AxisSpline, 87, 149, 745, 753, 894, 973
- RM_DirectTorque, 973
- RM_DirectVelocity, 973
- RM_DriveMode, 973
- RM_Feed, 899, 973
- RM_Gcode, 901, 973
- RM_GroupTrack, 904, 974
- RM_HomeOffset, 758, 974
- RM_Track, 759, 904, 974
- RS, 26, 322, 325, 974
- RTOI, 26, 134, 974
- RTOS, 19, 26, 361, 377, 974
- RTOV, 26, 385, 400, 553, 581, 974
- rychlá smyčka, 43
- S10F2, 25, 172, 974
- SAI, 25, 172, 173, 175, 974
- SAT, 26, 269, 974
- SC2FA, 26, 271, 974
- SCU, 26, 240, 247, 254, 278, 281, 974
- SCUV, 26, 240, 247, 254, 281, 974
- SEL, 178, 974
- SELHEXD, 26, 178, 179, 974

- SELOCT, 178, 179, 974
- SELQUAD, 178, 179, 974
- SELSOCT, 26, 378, 974
- SELU, 26, 285, 555, 974
- servoventil, 439
- SETPA, 19, 26, 78, 87, 383, 385, 399, 402, 974
- SETPB, 401, 404, 405, 974
- SETPI, 401, 404, 405, 599, 974
- SETPR, 19, 26, 78, 383, 401, 404, 405, 408, 974
- SETPS, 19, 26, 78, 383, 404, 405, 974
- SETPX, 26, 402, 404, 405, 974
- SG, 18, 195, 207, 588, 974
- SGEN, 446, 974
- SGENTX, 448, 974
- SGI, 207, 974
- SGSLP, 407, 411, 974
- SHIFTOCT, 26, 181, 974
- SHLD, 26, 183, 394, 974
- SILO, 19, 383, 409, 411, 974
- SILOS, 19, 383, 413, 974
- simulace
 - běh v reálném čase, 49
 - parametry, 49
- Simulink, 43, 49, 587
- SINT, 26, 160, 185, 974
- SLEEP, 49, 974
- SMHCC, 26, 287, 974
- SMHCCA, 26, 291, 974
- SMTP, 20, 579, 611, 974
- SOLNP, 575, 975
- SOPDT, 19, 26, 241, 248, 416, 450, 975
- součinitel relativního tlumení, 152
- SPIKE, 26, 175, 176, 186, 975
- SPLIT_DT, 975
- SQR, 18, 26, 100, 136, 975
- SQRT, 18, 26, 100, 129, 137, 975
- SR, 26, 323, 975
- SRTF, 50, 975
- SSW, 26, 188, 555, 588, 975
- STATELOAD, 52, 975
- STATESAVE, 52, 54, 975
- STEAM, 20, 579, 613, 975
- STMGEN, 452, 975
- STOR, 19, 26, 361, 380, 975
- STR2DT, 975
- STURB, 454, 975
- SUB, 18, 26, 100, 103, 138, 975
- SUBSYSTEM, 385, 400
- SubSystem, 73, 76, 79, 86, 91, 975
- SWR, 26, 189, 555, 975
- SWU, 26, 285, 298, 975
- SWVMR, 555, 975
- SYSEVENT, 56, 975
- SYSLOG, 58, 975
- T2STR, 975
- TASK, 17, 28, 35, 42, 48, 50, 59, 62, 418, 420, 975
- task
 - quick, 48
- TC, 19, 327, 332, 975
- TIME, 19, 327, 329, 334, 975
- TIMER, 18, 26, 301, 324, 975
- TIODRV, 42, 62, 975
- ToFile, 93, 975
- ToWorkspace, 94, 975
- trajektorie
 - časově optimální, 148
- TRIM, 26, 381, 975
- TRND, 31, 341, 356, 359, 975
- TRNDV, 341, 359, 975
- TS, 335, 337, 356, 975
- TS2NS, 337, 975
- TSE, 26, 278, 281, 299, 975
- typ
 - parametr, 23
 - výstup, 23
 - vstup, 23
- typy
 - proměnných, 23
- TZ2UTC, 975
- UART, 20, 579, 600, 616, 975
- UNIPi_CHANNEL, 944, 976
- UNIPi_PRODUCT, 946, 976
- UNIPi_S1AI, 947, 976

UNIP1_S1AOR, 948, 976
UNIP1_S1CNT, 949, 976
UNIP1_S1DI, 950, 976
UNIP1_S1DO, 951, 976
UNIP1_S1LED, 952, 976
UNIP1_S1PWM, 953, 976
UNIP1_S2AI, 954, 976
UNIP1_S2AO, 955, 976
UNIP1_S2CNT, 956, 976
UNIP1_S2DI, 957, 976
UNIP1_S2RO, 958, 976
UTC2TZ, 976
UTOI, 26, 139, 976

VDEL, 26, 190, 976
VIN, 25, 26, 82, 84, 90, 95, 597, 976
VOUT, 89, 97, 597, 976
VTOR, 26, 169, 556, 581, 897, 976

WASM, 976
WEEK, 976
WSCH, 19, 327, 338, 976
WWW, 64, 976

zásobník
 velikost, 40
ZV4IS, 26, 191, 976

